

پایان نامه کارشناسی ارشد در رشته ی مهندسی فناوری اطلاعات (تجارت الکترونیک)

ارائه یک مدل فرمال برای تحلیل زیر ساخت نرم افزاری در دیتا سنترها

منابع پایه ی پژوهش

چکیده

ارائه یک مدل فرمال برای تحلیل زیر ساخت نرم افزاری در دیتا سنترها

به کوشش

علی حسن بیگی

در این تحقیق با استفاده از زبان فرمال شبکه های پتری، از بخش ها و جنبه های مختلف لایه مجازی سازی در دیتا سنترها، مدل های فرمال متعددی تهیه شده است. سپس به کمک ابزار ریاضی، رفتار این مکانیزم ها برای تعیین سطح خوش رفتاری سیستم بررسی شده است. برای نیل به این هدف، در ابتدا ابزار مدل سازی فرمال اعم از جبری و مبتنی بر مدل بررسی و مقایسه شده است. سپس با تشریح تکنولوژی مجازی سازی در سیستم های کامپیوتری، یک دیتا سنتر نمونه کوچک مطابق استانداردهای دیتا سنترها طراحی شده و سپس در لایه های مختلف توسط شبکه های پتری مدل گردیده است. این مدل سازی از سطح کاملاً انتزاعی^۱ (لایه ۱) برای کل دیتا سنتر شروع و با طراحی مدل برای زیر سیستم ها و مکانیزم هایی مانند ساختار شبکه، ذخیره سازی و نیز سرویس های سطح بالا مانند HA و Fault Tolerance پایان می یابد. در پایان هر بخش، پس از ارائه و تشریح مدل، به تحلیل و ارزیابی آن و بررسی خصوصیات رفتاری سیستم از روی مدل فرمال آن

¹ Abstract

پرداخته شده است. در نهایت، نتیجه گیری از میزان خوش رفتاری بخش های مختلف این لایه در دیتا سنتر ارائه شده است.

منابع پایه اس پژوهش

فهرست مطالب

صفحه	عنوان
۱	فصل اول: مقدمه
۱	۱.۱. بیان مسئله و ضرورت تحقیق
۷	۱.۲. دامنه تحقیق
۷	۱.۳. پیشینه تحقیق
۸	۱.۴. اهداف تحقیق
۸	۱.۵. سوالات تحقیق
۹	۱.۶. مراحل و روش تحقیق
۱۰	۱.۷. دستاوردهای تحقیق
۱۰	۱.۸. نگاهی کلی بر پایان نامه
۱۲	فصل دوم: معرفی و بررسی ابزار فرمال برای مدل سازی سیستم ها
۱۴	۲.۱. Abstract State Machine (ASM)
۱۷	۲.۲. LOTOS
۱۸	۲.۲.۱. مقدمه ای بر جبر پروسه ها
۲۲	۲.۳. VDM-SL
۲۳	۲.۴. شبکه های پتری
۲۵	۲.۴.۱. خصوصیات رفتاری
۲۶	۲.۴.۱.۱. Reachability
۲۶	۲.۴.۱.۲. Boundedness
۲۶	۲.۴.۱.۳. Liveness
۲۸	۲.۴.۱.۴. Reversibility
۲۸	۲.۴.۱.۵. Coverability

۲۹	۲،۴،۲ زیر مجموعه های شبکه های پتری
۳۰	State Machine (SM) ۲،۴،۲،۱
۳۰	Marked Graph (MG) ۲،۴،۲،۲
۳۱	Free-choice net (FC) ۲،۴،۲،۳
۳۱	Extended Free-choice net (EFC) ۲،۴،۲،۴
۳۱	Asymmetric choice net (AC) ۲،۴،۲،۵
۳۲	۲،۴،۳ قضایا و فرضیات
۳۵	۲،۵ مقایسه و جمع بندی
۳۸	فصل سوم: بررسی معماری دیتا سنترها
۳۹	۳،۱ Microsoft Hyper-V
۴۲	۳،۱،۱ بررسی اجزاء معماری Hyper-V
۴۳	۳،۱،۱،۱ APIC
۴۳	۳،۱،۱،۲ Child Partition
۴۳	۳،۱،۱،۳ Hypercall
۴۳	۳،۱،۱،۴ Hypervisor
۴۳	۳،۱،۱،۵ IC
۴۴	۳،۱،۱،۶ I/O stack
۴۴	۳،۱،۱،۷ Root Partition
۴۴	۳،۱،۱،۸ VID
۴۴	۳،۱،۱،۹ VMBus
۴۴	۳،۱،۱،۱۰ VMMS
۴۴	۳،۱،۱،۱۱ VMWP
۴۵	۳،۱،۱،۱۲ VSC
۴۵	۳،۱،۱،۱۳ VSP
۴۵	۳،۱،۱،۱۴ WinHv
۴۵	۳،۱،۱،۱۵ WMI
۴۶	۳،۱،۲ نقاط ضعف
۴۶	۳،۲ Xen
۴۸	۳،۲،۱ بررسی اجزاء معماری Xen
۴۸	۳،۲،۱،۱ Xen Hypervisor
۴۸	۳،۲،۱،۲ Domain 0
۴۹	۳،۲،۱،۳ Domain U
۵۰	۳،۲،۱،۴ Xenstored
۵۰	۳،۲،۲ ارتباطات مابین دامنه صفر و دامنه های U

۵۰	VMware ESXi ۳,۳
۵۲	VMware ESX بررسی اجزاء معماری ۳,۳,۱
۵۲	VMkernel ۳,۳,۱,۱
۵۲	File System ۳,۳,۱,۲
۵۳	CIM ۳,۳,۱,۳
۵۴	طراحی یک معماری برای دیتا سنتر ۳,۴
۵۶	مختصری درباره vSphere ۳,۵
۵۸	سرویس های زیر ساختی vSphere ۳,۵,۱
۵۹	سرویس های کاربردی vSphere ۳,۵,۲
۵۹	سرور VMware vCenter ۳,۵,۲,۱
۵۹	Client ها ۳,۵,۲,۲
۵۹	ESX ۳,۵,۲,۳
۵۹	vCenter Server ۳,۵,۲,۴
۵۹	VMFS ۳,۵,۲,۵
۶۰	SMP ۳,۵,۲,۶
۶۰	VCMS ۳,۵,۲,۷
۶۰	VI Client ۳,۵,۲,۸
۶۰	VMware VMotion ۳,۵,۲,۹
۶۰	Storage VMotion ۳,۵,۲,۱۰
۶۱	VMware HA ۳,۵,۲,۱۱
۶۲	DRS ۳,۵,۲,۱۲
۶۳	Consolidated Backup ۳,۵,۲,۱۳
۶۳	vSphere SDK ۳,۵,۲,۱۴
۶۴	Fault Tolerance ۳,۵,۲,۱۵
۶۴	سرویس های توزیع شده در vSphere ۳,۵,۳
۶۴	معماری شبکه ۳,۵,۴
۶۶	معماری محل ذخیره سازی داده ها ۳,۵,۵
۶۹	معماری سرور مدیریت VirtualCenter ۳,۵,۶
۷۰	User Access Control ۳,۵,۶,۱
۷۱	Core Service ۳,۵,۶,۲
۷۱	جمع بندی ۳,۶
۷۲	فصل چهارم: ارائه مدل فرمال برای دیتا سنتر و تحلیل آن
۷۳	تشریح دیتا سنتر نمونه ۴,۱
۷۷	ارائه مدلی از رفتار کلی دیتا سنتر ۴,۲

۸۷	۴,۲,۱. ارزیابی و تحلیل مدل فرمال
۸۷	Liveness .۴,۲,۱,۱
	Safeness .۴,۲,۱,۲
	۸۷
۸۹	Reversibility .۴,۲,۱,۳
۹۰	۴,۲,۲. بررسی لایه های پایینتر مدل فرمال
۹۱	۴,۳. بررسی نحوه کار سرویس HA
۹۱	۴,۳,۱. شرح سرویس VMware HA
۹۵	۴,۳,۲. ارائه مدل فرمال از نحوه کار سرویس HA
۹۸	۴,۳,۳. تحلیل و ارزیابی مدل
۹۹	Liveness .۴,۳,۳,۱
	Safeness .۴,۳,۳,۲
	۱۰۱
۱۰۲	Reversibility .۴,۳,۳,۳
۱۰۶	۴,۴. بررسی نحوه کار سرویس Fault Tolerance
۱۰۶	۴,۴,۱. تشریح ساختار سرویس Fault Tolerance
۱۱۰	۴,۴,۲. ارائه مدل فرمال از نحوه کار سرویس Fault Tolerance
۱۱۱	۴,۴,۳. تحلیل و ارزیابی مدل
۱۱۲	Liveness .۴,۴,۳,۱
	Safeness .۴,۴,۳,۲
	۱۱۳
۱۱۳	Reversibility .۴,۴,۳,۳
۱۱۴	۴,۵. بررسی نحوه کار سرویس VMotion
۱۱۴	۴,۵,۱. تشریح ساختار سرویس VMotion
۱۱۸	۴,۵,۲. ارائه مدل فرمال از نحوه کار سرویس VMotion
۱۱۹	۴,۵,۳. تحلیل و ارزیابی مدل
۱۲۰	Liveness .۴,۵,۳,۱
	Safeness .۴,۵,۳,۲
	۱۲۱
۱۲۲	Reversibility .۴,۵,۳,۳
۱۲۴	۴,۶. بررسی ساختار داخلی ESX hypervisor
۱۲۴	۴,۶,۱. تشریح ساختار ESX

۱۲۵	۴,۶,۲. ارائه یک مدل فرمال از نحوه کار ESX
۱۲۹	۴,۶,۳. بررسی و تحلیل مدل Liveness. ۴,۶,۳,۱ ۱۲۹ Safeness. ۴,۶,۳,۲ ۱۳۱ Reversibility. ۴,۶,۳,۳
۱۳۲	۴,۷. تشریح ساختار سیستم ذخیره سازی در ESX
۱۳۳	۴,۷,۱. تشریح ساختار سیستم ذخیره سازی
۱۳۳	۴,۷,۲. ارائه یک مدل فرمال از نحوه کار سیستم ذخیره سازی در ESX
۱۳۸	۴,۷,۳. بررسی و تحلیل مدل Liveness. ۴,۷,۳,۱ ۱۴۴ Safeness. ۴,۷,۳,۲ ۱۴۵ Reversibility. ۴,۷,۳,۳
۱۴۴	۴,۸. معماری ساختار شبکه در ESX
۱۴۶	۴,۸,۱. تشریح ساختار شبکه
۱۴۸	۴,۸,۲. ارائه مدل فرمال برای ساختار شبکه در ESX
۱۴۸	۴,۸,۳. بررسی و تحلیل مدل Reversibility. ۴,۸,۳,۱ ۱۵۵ Liveness. ۴,۸,۳,۲ ۱۶۱ Safeness. ۴,۸,۳,۳ ۱۶۵ ۴,۹. معماری سوئیچ مجازی در ساختار شبکه
۱۴۶	۴,۹,۱. تحلیل ساختار سوئیچ مجازی
۱۴۸	۴,۹,۲. ارائه یک مدل فرمال برای سوئیچ مجازی
۱۶۹	۴,۹,۳. بررسی و تحلیل مدل Liveness. ۴,۹,۳,۱ ۱۷۲ Safeness. ۴,۹,۳,۲ ۱۷۳ Reversibility. ۴,۹,۳,۳
۱۷۲	۴,۱۰. جمع بندی

۱۷۶	فصل پنجم: نتیجه گیری و پیشنهادات
۱۸۰	مراجع
۱۸۶	پیوست: مجموعه کامل مدل های پتری طراحی شده در پایان نامه
۱۹۵	چکیده به زبان انگلیسی

فهرست جدول ها

صفحه	عنوان و شماره
۳۶	جدول ۲,۱. خلاصه مقایسه ابزار توصیف فرمال
۴۱	جدول ۳,۱. سیستم های عامل قابل پشتیبانی توسط Hyper-V R2
۴۷	جدول ۳,۲. سیستم های عامل قابل پشتیبانی توسط Xen نسخه ۳.
۱۶۴	جدول ۴,۱. گزارهای فعال در وضعیت های M_0 الی M_{15} از مدل پتری ۴,۳۰
۱۷۷	جدول ۵,۱. خلاصه نتایج به دست آمده از تحلیل رفتار زیر سیستم ها و سرویس های VMware ESX

فهرست شکل ها

صفحه	عنوان
۴	شکل ۱,۱. نحوه قرارگیری لایه های نرم افزاری بر روی سرور
۵	شکل ۱,۲. شمای کلی دیتا سنتر با معماری مجازی
۱۳	شکل ۲,۱. مثال هایی از زبان های فرمال و تقسیم بندی آنها [۱۰]
۱۴	شکل ۲,۲. روال طراحی یک سیستم نمونه به کمک زبان های فرمال [۱۰]
۱۸	شکل ۲,۳. نقطه gate در جبر پروسه ها [۴]
۱۹	شکل ۲,۴. مدل تولید کننده- مصرف کننده به کمک LOTOS [۱۷]
۲۵	شکل ۲,۵. مثالی از مدل سازی یک پروتکل به کمک شبکه های پتری [۲۳]
۲۷	شکل ۲,۶. نمونه ای از شبکه پتری non-Live [۲۳]
۳۰	شکل ۲,۷. نشانه گذاری برای: a) مجموعه موقعیت های ورودی و خروجی برای t و b) مجموعه گزاره های ورودی و خروجی برای p [۲۳]
۳۲	شکل ۲,۸. مثال هایی از زیر مجموعه های شبکه های پتری [۲۳]
۳۳	شکل ۲,۹. یک شبکه پتری و گراف نشانه دار مربوط به آن [۲۳]
۴۲	شکل ۳,۱. معماری سطح بالای Hyper-V [۴۰]
۴۷	شکل ۳,۲. شمایی از معماری Xen [۵۱]
۴۹	شکل ۳,۳. نحوه سرویس دهی به ماشین میزبان توسط Qemu-DM [۵۱]
۵۲	شکل ۳,۴. شمایی از معماری VMware ESXi [۳]
۵۴	شکل ۳,۵. ساختار شماتیک مدیریت CIM [۳]

- ۵۸ شکل ۳,۶. مثالی از مفاهیم میزبان، کلاستر و مخزن منابع [۶۰]
- ۶۲ شکل ۳,۷. طرز کار سرویس HA [۶۱]
- ۶۵ شکل ۳,۸. شمایی از معماری شبکه در محیط مجازی [۶۰]
- ۶۶ شکل ۳,۹. شمایی از معماری ذخیره سازی [۶۰]
- ۶۸ شکل ۳,۱۰. طرز کار RDM [۶۰]
- ۷۰ شکل ۳,۱۱. شمایی از ساختار سرور مدیریت VirtualCenter [۶۰]
- ۷۳ شکل ۴,۱. زیر ساخت دیتا سنتر مجازی [۶۳]
- ۷۵ شکل ۴,۲. یک الگوی نمونه برای زیر ساخت دیتا فیزیکی سنتر [۶۰]
- ۷۶ شکل ۴,۳. ساختار شماتیک دیتا سنتر نمونه
- ۷۸ شکل ۴,۴. مدل پتری طراحی شده برای دیتا سنتر نمونه
- ۸۸ شکل ۴,۵. گراف پوشا برای مدل پتری شکل ۴,۴
- ۸۹ شکل ۴,۶. نتیجه تحلیل فضای حالت به وسیله نرم افزار PIPE
- ۹۶ شکل ۴,۷. مدل پتری نحوه کار سرویس HA
- ۱۰۰ شکل ۴,۸. عضویت شبکه ۴,۷ در زیرکلاس های شبکه های پتری
- ۱۰۱ شکل ۴,۹. گراف پوشای مدل ۴,۷
- ۱۰۲ شکل ۴,۱۰. نتیجه تحلیل فضای حالت بر روی مدل
- ۱۰۳ شکل ۴,۱۱. نحوه توزیع توکن در وضعیت S8
- ۱۰۴ شکل ۴,۱۲. شبیه سازی شبکه پتری شکل ۴,۷
- ۱۰۵ شکل ۴,۱۳. چندین نمونه از شبیه سازی اجرای شبکه پتری
- ۱۱۰ شکل ۴,۱۴. مدل پتری نحوه کار سرویس Fault Tolerance
- ۱۱۲ شکل ۴,۱۵. نحوه توزیع توکن ها در M_3 و M_4
- ۱۱۴ شکل ۴,۱۶. گراف پوشای مدل پتری شکل ۴,۱۴
- ۱۱۸ شکل ۴,۱۷. مدل پتری نحوه کار سرویس VMotion
- شکل ۴,۱۸. گراف نشانه دار (G, M_0) مربوط به مدل ۴,۱۷
- ۱۲۰
- ۱۲۳ شکل ۴,۱۹. گراف پوشای مدل ۴,۱۷

- ۱۲۵ شکل ۴,۲۰. مدل پتری طرز کار ESX
- ۱۳۰ شکل ۴,۲۱. گراف جهت دار معادل شبکه پتری ۴,۲۰
- ۱۳۲ شکل ۴,۲۲. گراف پوشای کدل پتری ۴,۲۰
- ۱۳۴ شکل ۴,۲۳. نمای شماتیک مدل چند لایه ای سیستم ذخیره سازی در ESX [۶۳]
- ۱۳۹ شکل ۴,۲۴. مدل پتری ارائه شده از نحوه کار سیستم ذخیره سازی در ESX
- ۱۴۴ شکل ۴,۲۵. گراف جهت دار متناظر با مدل پتری ۴,۲۴
- ۱۴۶ شکل ۴,۲۶. گراف پوشای شبکه پتری ۴,۲۴
- ۱۴۷ شکل ۴,۲۷. نرم افزار تحلیلگر گراف، در حال اجرای الگوریتم اول عمق
- ۱۴۹ شکل ۴,۲۸. نحوه ارتباط کارت شبکه مجازی و سوئیچ مجازی [۶۲]
- ۱۵۳ شکل ۴,۲۹. شمای کلی از ساختار شبکه در سرور ESX
- ۱۵۶ شکل ۴,۳۰. مدل پتری تهیه شده از ساختار شبکه در ESX
- ۱۶۲ شکل ۴,۳۱. گراف پوشای مدل پتری شکل ۴,۳۰
- ۱۶۳ شکل ۴,۳۲. جستجوی اول عمق گراف شکل ۴,۳۱
- ۱۷۰ شکل ۴,۳۳. مدل فرمال از نحوه کار سوئیچ مجازی
- ۱۷۳ شکل ۴,۳۴. گراف جهت دار متناظر با مدل پتری ۴,۳۳
- ۱۷۴ شکل ۴,۳۵. گراف پوشای مدل پتری ۴,۳۳

فصل اول: مقدمه

۱.۱ بیان مسئله و ضرورت تحقیق

نیاز بشر به پردازش و ذخیره سازی اطلاعات در دهه های گذشته همواره رشد صعودی و شتابدار داشته است. به گونه ای که حرکت از سیستم های توزیع شده بر روی سوپر کامپیوترهای گران قیمت به شبکه های بسیار پر قدرت و ارزان در مدت نسبتاً کوتاهی صورت گرفته است. همچنین نیاز به مدیریت اطلاعات، پردازش، گردش کار و دیگر ابزار مدیریتی همواره رشد فزاینده داشته است. به طبع این نیاز، ساختار سیستم های کامپیوتری در سطوح فنی و مدیریتی نیز رشد کرده و پیچیده تر شده است.

به منظور جوابگویی به این حجم فزاینده درخواست ها و نیاز بازار به منابع پردازش و ذخیره سازی اطلاعات و نیز به منظور ارائه سرویس های مورد نیاز با کیفیت مناسب و قابل رقابت، یکی از بهترین راه های پیشنهاد شده، متمرکز نمودن این منابع و مدیریت صحیح آنها است. به این منظور و برای به حداکثر رساندن کیفیت خدمات و حداقل نمودن هزینه ها یکی از رایج ترین راهکارهای موجود راه اندازی مراکز داده یا دیتا سنتر ها می باشد. در این طرح با آماده سازی زیر ساخت های فیزیکی، امنیتی، شبکه ای، سخت افزاری و نرم افزاری، مجموعه ای از سرورهای قدرتمند برای ارائه سرویس های مورد نیاز مشتریان در نظر گرفته می شود. این سرورها با خطوط بسیار پر سرعت بر حسب نیاز به اینترنت یا شبکه های سازمانی متصل می گردند و با نصب سیستم های عامل و نرم افزارها و سرویس های مورد نیاز به کاربران خدمات لازم را ارائه می نمایند. با وجود چنین مراکزی دیگر

سازمان ها و مراکز تجاری، صنعتی، دانشگاهی و غیره نیازی به راه اندازی مراکز سرویس دهی محلی^۱ و نیز متحمل شدن هزینه های نگهداری، به روز رسانی و استخدام متخصصین نخواهند داشت. در ادامه به بررسی اجمالی دیتا سنترها خواهیم پرداخت تا بتوانیم طرح پیشنهادی را تشریح نمائیم.

تعریف دیتا سنتر: مجموعه ای از سیستمهای پشتیبانی (از جمله زیر ساخت سخت افزاری passive، زیرساخت خنک کننده، زیر ساخت تامین انرژی، اطفاء حریق و غیره)، منابع پردازشی سخت افزاری شامل سرورها، تجهیزات زیرساخت شبکه، زیرساخت ذخیره سازی داده ها و زیرساخت نرم افزاری شامل ابزار یک پارچه سازی^۲ مجموعه ای از سیستم های عامل، مجموعه ای از نرم افزارهای کاربردی شامل سرویس ها، تعدادی پایگاه داده، مجموعه ای از ابزارهای امنیتی نرم افزاری و سخت افزاری و یک ساختار مدیریتی است. این سیستم به کمک خطوط پرسرعت به شبکه های خارجی (Extranet, Intranet یا اینترنت) متصل است [۱].

با توجه به رشد نیازها و احتیاج کاربران به انعطاف پذیری و تحمل خطای بالا در این مراکز پردازشی، در سال های اخیر تکنولوژی مجازی سازی^۳ به عنوان پاسخی به این نیازها و بهترین شیوه یکپارچه سازی ارائه شده و بسیار رشد کرده است. در حقیقت، این تکنولوژی به عنوان لایه مدیریت نرم افزاری و سیستم عاملی دیتا سنتر مورد استفاده قرار می گیرد. در ادامه به تشریح تکنولوژی مجازی سازی و نحوه استفاده از آن در این طرح خواهیم پرداخت.

تکنولوژی مجازی سازی یکی از جوانترین نظریه های مطرح شده در علم کامپیوتر می باشد که در ده سال اخیر توجه زیادی را به خود جلب نموده است. این تکنولوژی از این بابت بسیار جذاب است که انعطاف پذیری و امکانات خارق العاده ای را بر روی همان بستر سخت افزاری موجود ارائه می دهد و استفاده از آن هزینه بسیار ناچیزی برای سازمان دارد.

معماری مجازی سازی، همه منابع پردازشی از جمله سرورها، منابع ذخیره سازی^۴ و شبکه را به یک ساختار مجازی نگاشت می دهد. این زیر ساخت با گردآوری همه منابع و نمایش مجموعه ای

¹ Local Farm Servers

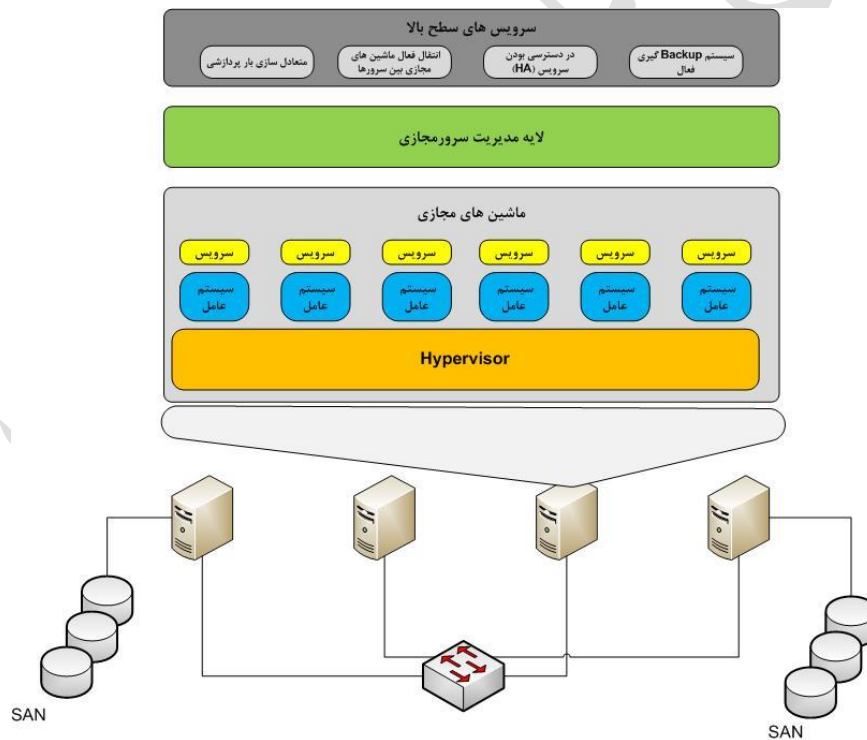
² Consolidation

³ Virtualization technology

⁴ Storage

ساده شده و یکپارچه از آنها، مدیر را در درک بهتر ساختار فنی دیتا سنتر و مدیریت و تغییر آن یاری می‌رساند. به کمک این ساختار می‌توان منابع توزیع شده در یک دیتا سنتر را به صورت مجموعه ای یکپارچه از ابزار مدیریت نمود. همچنین می‌توان از دیتا سنتر برای مصارف گوناگونی استفاده کرد بدون اینکه نگران گوناگونی سخت افزارها و نحوه اتصال آن‌ها به سیستم باشیم؛ [۲] و [۳].

از این تکنولوژی برای طراحی زیر ساخت نرم افزاری دیتا سنتر استفاده خواهد شد. با این توضیح که به جای نصب یک سیستم عامل بر روی هر دستگاه سرور، از یک نرم افزار مجازی سازی به نام Hypervisor استفاده می‌شود. این نرم افزار شبه سیستم عامل به مدیر سیستم اجازه می‌دهد که به تعداد دلخواه کامپیوتر مجازی^۱ بر روی سرور مذکور راه اندازی کرده و سیستم عامل و سرویس های دلخواه را بر روی آن نصب نماید (شکل ۱،۱).

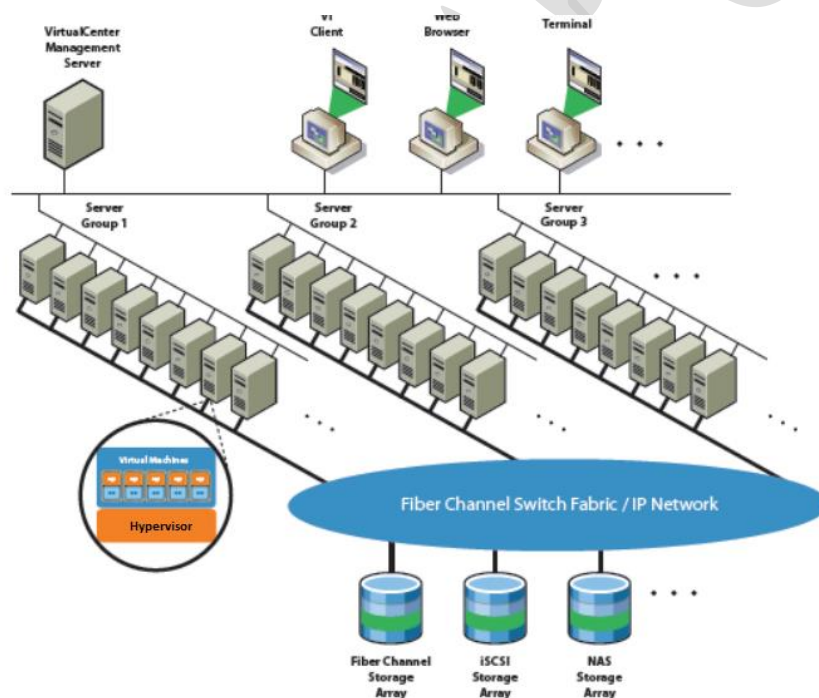


شکل ۱،۱. نحوه قرارگیری لایه های نرم افزاری بر روی سرور

^۱ Virtual Machine

با این ترکیب می توان امکانات بسیار زیادی از جمله قابلیت دسترسی دائمی به سرویس ها (HA)^۱ و مقاوم سازی سرویس ها در مقابل خطا^۲ که از ضروریات چنین دیتا سنتری می باشد را با کمترین هزینه میسر نمود. همچنین امکان انتقال این کامپیوترهای مجازی در حال کار از روی یک سرور به سرور دیگر را بدون تاخیر زمانی وجود دارد.^۳

در دیتاسنتری با این ابعاد، اغلب سرویس های در حال کار بسیار حیاتی و حساس بوده و از کار افتادن آن ها هزینه های هنگفت و بعضا جبران ناپذیری برای سازمان مربوطه به دنبال خواهد داشت. به همین دلیل لازم است امکانات حرفه ای را در دیتا سنتر به منظور محافظت از سرویس ها پیاده سازی نمائیم تا در دسترس بودن و سلامت آنها را تضمین کند. شکل ۱،۲ شمایی کلی از یک دیتا سنتر را با استفاده از معماری یاد شده نشان می دهد.



شکل ۱،۲. شمای کلی دیتا سنتر با معماری مجازی

^۱ High Availability

^۲ Fault Tolerance

^۳ VMotion

با توجه به نیاز به این مراکز و پیچیدگی ذاتی آنها، ترسیم یک مدل فرمال از ماهیت یک دیتا سنتر، چه پیش از طراحی^۱ و چه پس از آن^۲، می تواند در شناخت طرز کار و چگونگی فعالیت چنین مرکزی نقش به سزایی داشته باشد. از جمله این کاربردها می توان به تشخیص بن بست ها^۳ و گلوگاه ها^۴ قبل از طراحی و محک زدن^۵ سیستم بعد از طراحی اشاره نمود. با در دست داشتن این مدل (تصویر فرمال) جریان کنترل در سیستم قابل رویت بوده و در نتیجه رفتار سیستم را می توان بررسی و پیش بینی نمود [۴]. البته باید توجه داشت که در سیستم های واقعی از جمله دیتا سنترها، به دست آوردن مدل جامع تقریباً غیر ممکن بوده و تنها می توان بخش هایی از سیستم را با نادیده گرفتن برخی از پارامترها مدل نمود. هرچقدر مدل به سیستم واقعی نزدیکتر باشد بررسی رفتار سیستم به کمک مدل حاصل دقیقتر و کاربردی تر خواهد بود. در بخش های بعدی با بررسی دقیقتر ماهیت مدل سازی فرمال، با انواع شیوه ها در این حوزه^۶ بیشتر آشنا خواهیم شد.

به طور کلی متد های فرمال نوع خاصی از شیوه های بیان فرمال مسائل هستند که از آنها برای تشریح و تبیین^۷ سیستم های کامپیوتری و همچنین اثبات رفتار آنها^۸ در سطح سخت افزار و نرم افزار استفاده می شود. هدف از توضیح رفتار یک سیستم به کمک روش های فرمال، بررسی رفتار و خصوصیات سیستم از جمله میزان حد پذیری^۹، بازگشت پذیری^{۱۰} و نیز پارامترهای انتزاعی تر مانند میزان ثبات^{۱۱} و پایداری^{۱۲} می باشد [۵].

بدیهی است انجام چنین کاری در مورد سیستم های واقعی با توجه به پارامترهای متعدد و ساختار پیچیده آنها بسیار وقت گیر و دشوار است و در بسیاری از مواقع فقط بخش هایی از سیستم را می توان در حد قابل قبولی تشریح و مدل نمود. به همین دلیل و نیز به دلیل هزینه بسیار گزاف این

¹ Pre-analysis modeling

² Post-analysis modeling

³ Deadlock

⁴ Bottleneck

⁵ Benchmarking

⁶ Formal Modeling

⁷ Specification

⁸ Verification

⁹ Boundedness

¹⁰ Reversibility

0

¹¹ Robustness

1

¹² Stability

2

فرایند، استفاده از شیوه های فرمال برای توضیح رفتار سیستم فقط در مورد سیستم های بسیار حساس و گران قیمت صورت می گیرد.

در این تحقیق، از زبان شبکه های پتری که ابزاری گرافیکی برای تشریح رفتار سیستم ها می باشد بهره گرفته شده است. این زبان در واقع نوع خاصی از ماشین های متناهی (اتوماتا) می باشد که امکان ترسیم جریان کنترل در سیستم را به صورت ساختار گراف و تعریف مجموعه ها فراهم می کند.

۱.۲. دامنه تحقیق

در این تحقیق، به کمک زبان فرمال شبکه های پتری، مدل فرمالی از بخش های مختلف یک دیتا سنتر نمونه ارائه خواهد شد. به این منظور در ابتدا ساختار دیتا سنترهای نوین شرح داده شده و سپس یک مدل کلی و انتزاعی برای آن ارائه می گردد. در مراحل بعد، هریک از بخش های دیتا سنتر نمونه در لایه های بعدی طراحی و مدل خواهد شد. این بخش ها عبارتند از: سرویس HA، سرویس Fault Tolerance، سرویس VMotion، ساختار ESX، معماری سیستم ذخیره سازی در ESX، معماری شبکه در ESX و معماری سوئیچ مجازی در ساختار شبکه.

در انتهای هر بخش، مدل های پتری طراحی شده به کمک ابزار و متد های فرمال بررسی و ارزیابی خواهند شد. بخشی از این متدها مرتبط با نوع و گروه بندی شبکه هایی است که مدل مذکور در آن جای می گیرد، و بخشی نیز وابسته به خواص عمومی گراف ها می باشد.

۱,۳. پیشینه تحقیق

با توجه به قابلیت بسیار زیاد شبکه های پتری در مدل سازی سیستم های موازی، همگام، توزیع شده، غیرتعینی، و آماری، می توان نمونه های بسیاری از توصیف های فرمال سیستم های واقعی را به کمک این ابزار ذکر کرد. از آن جمله می توان به موارد زیر اشاره کرد.

- طراحی مدل شبکه ترافیک شهری به کمک شبکه های پتری رنگی (سطح بالا) برای تخمین گلوگاه ها و نیز میزان کارامدی سیستم ترافیک [۶].
- مدل سازی فرمال یک خط اولید به همراه پنج ربات برای تحلیل خصوصیات رفتاری خط تولید و میزان خوش رفتاری آن [۷]
- مدل سازی یک پردازنده مبتنی بر VLSI که در یک سوپر کامپیوتر به کار رفته است. این مدل سازی به کمک شبکه های پتری رنگی و در سطح رجیستر های این ریز تراشه انجام شده است.

در مرجع [۸] می توانید لیست بلندی از سیستم های صنعتی مدل شده با شبکه های پتری را که به صورت پروژه های مجزا انجام شده مشاهده نمائید. با این حال در زمینه لایه نرم افزاری و مجازی سازی در دیتا سنترها و نیز بخش سیستم عاملی آن تا به حال مدل سازی جامعی به کمک شبکه های پتری انجام نشده است. هر چند تلاش هایی در زمینه مدل سازی زیر ساخت سخت افزاری دیتا سنتر و برای بهینه نمودن مصرف انرژی در آنها و طراحی دیتا سنترهای سبز (دوستدار محیط زیست) با شبکه های پتری انجام شده است. [۹]

۱,۴. اهداف تحقیق

- تشریح ساختار نرم افزاری دیتا سنترهای نوین

- تشریح ماهیت و مکانیزم های مجازی سازی، فواید و کاربرد های آن و نحوه طراحی دیتا سنتر به کمک این تکنولوژی
- ارائه مدل های فرمال از ساختار های پیشنهاد شده در بخش های نرم افزاری دیتا سنتر بر روی پلتفرم مجازی.
- ارائه تحلیل و اثبات خواص رفتاری سیستم بر روی مدل های طراحی شده.

۱,۵. سوالات تحقیق

- موارد زیر می تواند بخشی از سوالاتی باشد که در این تحقیق مطرح می شوند:
- ساختار سیستم عاملی و نرم افزاری یک دیتا سنتر چگونه است؟
 - تکنولوژی مجازی سازی چگونه در ساختار نرم افزاری دیتا سنتر ترکیب می شود و کار می کند؟
 - چگونه می توان یک تصویر فرمال از یک دیتا سنتر نمونه که بر مبنای تکنولوژی گفته شده طراحی شده است را به دست آورد؟
 - چه دانشی می توان از مدل فرمال طراحی شده استخراج نمود؟

۱,۶. مراحل و روش تحقیق

با توجه به اینکه مرحله نهایی در تحقیق، رسیدن به مدل هایی فرمال است که ساختار داخلی سیستم و رفتار آن را به روشنی نشان دهد، باید در ابتدا مقدماتی از جمله مبانی نظری دیتا سنترها، مجازی سازی و نیز ابزار فرمال برای مدل سازی سیستم ها مطرح و تشریح گردد.

با این رویکرد، پس از تشریح ابزار فرمال و نیز انواع تکنولوژی های مورد استفاده در دیتا سنتر، با ترکیب این دو حوزه، به بررسی و تشریح ساختار داخلی و نحوه رفتار دیتا سنترها به کمک ابزار و زبان های مورد استفاده می پردازیم. در این مسیر دنبال کردن سوالات و بخش های زیر راهگشا خواهد بود:

۱. مدل سازی چیست و ابزار فرمال برای این کار چه هستند؟
 ۲. مدل سازی فرمال یک سیستم چه امکاناتی در اختیار ما قرار می دهد و دانش قابل استخراج از آن چیست؟
 ۳. دیتا سنترهای امروزی از چه بخش هایی تشکیل شده اند و چگونه قابل تجزیه هستند؟
 ۴. چگونه می توان به شیوه سلسله مراتبی، مدل های فرمالی از بخش های سیستم از کل به جزء طراحی نمود؟
- در پایان با ارزیابی مدل های تهیه شده، می توان به ارزیابی از میزان خطاپذیری و نقاط ضعف سیستم دست یافت.
- در اینجا ذکر این نکته ضروری است که در مدل سازی فرمال لایه مجازی سازی در سیستم های پردازشی، آنگونه که در این تحقیق به آن پرداخته شده، تحقیقات بسیار اندکی انجام شده است. از این رو بخش های زیادی در مورد پیشینه تحقیق به چشم نمی خورد.

۱,۷. دستاوردهای تحقیق

حاصل کار این تحقیق، مدل های فرمال متعددی از بخش های نرم افزاری یک دیتا سنتر نمونه است. این مدل ها می توانند در شناخت ضعف ها، ناکارآمدی ها و گلوگاه های سیستم نقشی فعال و محسوس بازی نماید. همچنین به کمک این مدل در زمان طراحی سیستم شناخت دقیقتری نسبت به کارکرد و رفتار آنها حاصل خواهد شد و همین امر به طراحی یک سیستم پایدار و دقیق کمک شایانی می نماید.

۱,۸. نگاهی کلی بر پایان نامه

این پایان نامه از بخش های زیر تشکیل شده است:

فصل دوم: بررسی و مقایسه زبان های فرمال برای مدل سازی، توصیف و بررسی سیستم های موازی، غیر همگام^۱، توزیع شده، غیر قطعی^۲، آماری^۳ و دینامیک. در این بخش توانایی ها و قابلیت های این ابزار با یکدیگر مقایسه و در نهایت یکی از آنها برای استفاده در بخش های بعد انتخاب می گردد.

فصل سوم: بررسی زیرساخت نرم افزاری دیتا سنتر بر مبنای تکنولوژی مجازی سازی. در این بخش همچنین امکانات و قابلیت های استفاده از این تکنولوژی در طراحی معماری دیتا سنتر مورد بررسی و تشریح قرار می گیرد.

فصل چهارم: ارائه مدل از بخش های مختلف معماری نرم افزاری دیتا سنتر و تحلیل مدل. در این بخش پس از طراحی مدل پتری هریک از ماژول ها، با تحلیل فرمال مدل متناظر، خصوصیات رفتاری مختلفی از آنها بررسی و اثبات می شوند.

¹ Asynchronous

² Nondeterministic

³ Stochastic

فصل دوم: مروری بر ابزار فرمال مدل سازی سیستم ها

طبق آمار سایت های غیر رسمی^۱ در حدود ۱۰۰ متد فرمال به صورت زبان، notation و ابزار وجود دارد که به منظور تشریح فرمال سیستم ها و مسائل از آنها استفاده می شود. پارامترهای زیادی وجود دارد که نقاط قوت و ضعف هر یک از ابزار مذکور را نسبت به دیگران تعیین می نماید اما در این بین تعدادی از این مصادیق به دلایلی عمومیت بیشتری یافته اند و بیشتر در مورد بررسی و به روز رسانی قرار گرفته اند.

به طور کلی می توان ابزار توصیف فرمال^۲ را به دو گروه متدهای جبری و متدهای مبتنی بر مدل تقسیم بندی نمود [۵] و [۱۰]. اگر فرض نمائیم که بهترین شیوه شناخت و توصیف سیستم ها شکستن آنها به بخش های کوچکتر و ساده تر^۳ برای کنترل پیچیدگی سیستم است، از بررسی شیوه توصیف هریک از این دو نوع ابزار می توان نتیجه گرفت که متدهای جبری برای توصیف interface بین بخش های سیستم مذکور مناسب است زیرا خصوصیات کارکردی^۴ یا ساختاری^۵ سیستم را به نحو مطلوبی نشان می دهد. با این حال این ابزار برای تشریح رفتار سیستم چندان کارآمد نیست. به منظور توصیف خصوصیات رفتاری یک سیستم^۶ از ابزار مدل سازی فرمال استفاده می گردد. در شکل

^۱ <http://formalmethods.wikia.com/wiki/VL#WikiaArticle>

^۲ Formal Specification

^۳ Component

^۴ Functional Properties

^۵ Structural properties

^۶ Behavioral Properties

۲,۱ مثالهایی از گروه های گفته شده و نیز تقسیم بندی توانایی آنها در تشریح سیستم های تسلسلی^۱ و موازی^۲ آورده شده است.

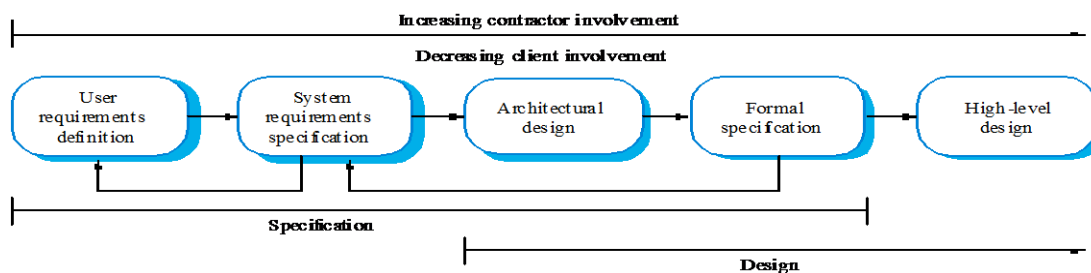
	Sequential	Concurrent
Algebraic	Larch (Guttag et al., 1993) , OBJ (Futatsugi et al., 1985)}	Lotos (Bolognesi and Brinksma, 1987)},
Model-based	Z (Spivey, 1992)} VDM (Jones, 1980)} B (Wordsworth, 1996)}	CSP (Hoare, 1985)} Petri Nets (Peterson, 1981)}

شکل ۲,۱. مثال هایی از زبان های فرمال و تقسیم بندی آنها [۱۰]

به طور کلی استفاده از توصیف فرمال سیستم ها قبل از طراحی و پیاده سازی، همراه با مشکلاتی مانند هزینه بر بودن، زمان بر بودن و نیز دشواری و پیچیدگی در بیان و ارزیابی آن است. این موانع باعث شده است تا این شیوه توسعه برخلاف آنچه که در دهه ۱۹۸۰ پیش بینی می شد، پرکاربرد ترین شیوه توسعه سیستم های نرم افزاری و سخت افزاری نشود. با این حال مزایای این روش پیاده سازی که کاهش محسوس خطاهای سیستم در هنگام طراحی و پیاده سازی است باعث شده است که در طراحی سیستم های حیاتی و گران قیمت مورد استفاده قرار گیرد. با توجه به اینکه توصیف یک سیستم به شکل تفکیک ناپذیری در طراحی آن گره خورده است، می توان برای طراحی سیستم به شیوه توصیف فرمال آن، روالی طبق شکل ۲,۲ را مد نظر قرار داد.

¹ Sequential

² Concurrent



شکل ۲.۲. روال طراحی یک سیستم نمونه به کمک متد های فرمال [۱۰]

همانگونه که دیده می شود، برقراری درک دقیقی از نیازمندیهای سیستم^۱، طراحی معماری و نیز توصیف (درک) فرمال سیستم به شکلی با یکدیگر ادغام شده اند که بازخورد هر مرحله از توصیف به کامل تر شدن طراحی در همان مرحله کمک می کند. بنابراین در پایان فاز توصیف فرمال، حداقل میزان خطا و حالات پیش بینی نشده در سیستم وجود خواهد داشت. درصد این خطا مستقیماً به میزان توانایی ابزار توصیف در بررسی و تبیین جنبه های مختلف کارکردی و رفتاری سیستم باز می گردد [۱۱].

در ادامه سعی می کنیم تعدادی از این ابزار فرمال را تشریح کرده و با هم مقایسه نمائیم. بعضی از نمونه ها تنها به صورت متد بوده و بعضی دیگر دارای گرامر نیز می باشند و زبان هستند.

۲.۱. ASM

ASM^۲ از ساختارهای کلاسیک ریاضی برای توصیف فرمال سیستم استفاده می کند بنابراین بسیار دقیق است. با این توضیح که ASM متدی مبتنی بر جبر رابطه ای و مناسب برای توصیف سیستم های تسلسلی است. باید توجه داشت که اگر در توصیف مسائل، semantic متد به کار رفته مبهم باشد توصیف حاصل واضح تر از تعریف اولیه سیستم نخواهد بود. بنابراین ASM با یک تعریف

^۱ system requirements specification

^۲ Abstract State Machine

ریاضی، وضوح و یگانگی درک مفهوم مورد نظر را تضمین می کند.^۱ ASM شکل خاصی از finite-state automata است که از مجموعه ای از ساختار های داده ای و توابع ریاضی تشکیل شده است و برای توصیف الگوریتم ها بر اساس حالات آنها به کار می رود [۱۲].

در واقع ASM یک ساختار قراردادی ساده یا پیچیده از قواعد ریاضی است که از مجموعه ای از دامنه ها (یا مجموعه ها که هر یک نوع مشخصی از اشیا را نشان می دهند) و توابع و روابط قراردادی بر روی آنها تعریف شده اند [۱۳]. همانگونه که دیده می شود این مجموعه بسیار انتزاعی و مستقل از نوع مساله است.

دستیابی به سطوح بالایی از انتزاع از مهمترین مزایای ASM است.

در ادامه نمونه از تعریف فرمال ASM را برای سیستم PVM^۴ مشاهده می کنیم [۱۲]:

(1) *master : HOST*

مشخص کننده ی ماشینی است که PVM روی آن اجرا می شود و نقش کلیدی در مدیریت و نگهداری پیکره پویای PVM دارد.

(2) *arch : HOST → ARCH*

تابع arch که معماری هر یک از HOST ها را مشخص می کند. این معماری عضوی از مجموعه انتزاعی ARCH است.

(3) *daemon : HOST → DAEMON*

این تابع برای هر عضو از مجموعه HOST، پروسه ای از مجموعه DAEMON را مشخص می کند که وظیفه مدیریت برنامه های آن هاست و نیز ارتباطات بین پروسه ها را برعهده دارد. این daemon به کمک مجموعه ای از پروسه ها بر روی host پیاده سازی می شود.

(4) *demiurge : DAEMON*

¹ Well-understoodness

² Data Structure

³ States

⁴ Parallel Virtual Machine

daemon خاصی است که بر روی master host اجرا می شود (می توان آنرا به صورت daemon(master) = demiurge بیان کرد) و وظیفه آن ایجاد و حذف hostها و نگهداری ارتباطات آنهاست.

این اشیا و روابط انتزاعی مقدمه ای برای مدل ASM از VPM است. در حال حاضر اطلاعات بیشتری در مورد ماهیت hostها، daemonها و معماری آنها مورد نیاز نیست. در این زبان می توان از سودو کدهای ساده استفاده نمود بنابراین نحو^۱ زبان برای اکثر افراد قابل درک است.

بر خلاف بسیاری از ابزار، می توان مدل طرح شده به ASM را مستقیماً اجرا نمود تا نتیجه رفتار سیستم مشخص گردد. همچنین از این ابزار می توان برای مدل کردن طیف وسیعی از سیستم ها استفاده کرد. از جمله می توان به سیستم های تسلسلی، موازی، توزیع شده، real time، سیستم های متناهی و نامتناهی و غیره اشاره نمود که نشان دهنده انعطاف پذیری ابزار می باشد. در ادامه نمونه ای از نحو زبان ASM آورده شده است. این بخش چگونگی انتقال پیام مابین Agent ها را نشان می دهد. [۱۴].

```
class COMMUNICATOR
  Program() =
    let availableMsgs = {m | m in me.mailbox where
      ReadyToDeliver(m)}
    let selectedMsgs = chooseSubset(availableMsgs)

    forall msg in selectedMsgs
      me.mailbox(msg) := false           //delete the message
      let resolvedMsgs = ResolveMessage(msg) //resolve the message
      forall m in resolvedMsgs
        let a = Recipient(m)
        if a <> undef then           // if recipient found
```

¹ Syntax

<i>InsertMessage(a,m)</i>	<i>// forward the message</i>
<i>else</i>	
<i>skip</i>	<i>// else ignore message</i>

برای مطالعه بیشتر به [۲۹] الی [۳۲] مراجعه کنید.

۲.۲. LOTOS

^۱ LOTOS توسط سازمان ISO برای بیان فرمال استاندارد های مربوط به OSI ساخته شد. این زبان به عنوان به عنوان یک FDT^۲ تحت استاندارد ISO/IEC 8807 در سال ۱۹۸۹ ارائه گردید.^۳

LOTOS یک زبان توصیف فرمال مبتنی بر روابط جبری است که برای توصیف سیستم های موازی و توزیع شده مناسب می باشد [۱۵]. در واقع دو بخش اصلی تشکیل دهنده LOTOS عبارتست از ADT^۴ و جبر پروسه ها. به دلیل پشتیبانی از گونه های داده ای انتزاعی، این زبان می تواند تبادل اطلاعات و در نتیجه رابطه بین اجرا را به خوبی توصیف نماید. همچنین به دلیل جبری بودن، خواص عملکردی سیستم را توصیف می نماید. به علاوه، پشتیبانی از عملگرهای temporal باعث می شود که این زبان توانایی توصیف خواص رفتاری و جنبه های زمانی سیستم ها را نیز داشته باشد.

^۱ Language Of Temporal Ordering Specification

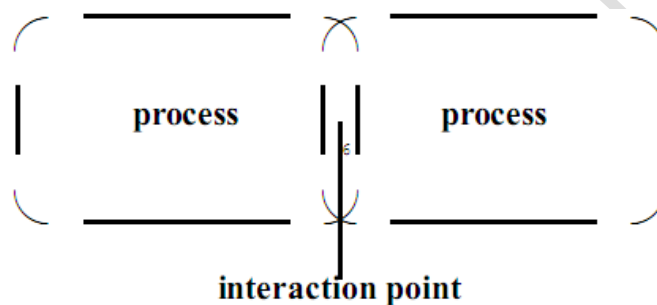
^۲ Formal Description Technique

^۳ http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16258

^۴ Abstract Data Type

۲,۲,۱. مقدمه ای بر جبر پروسه ها [۱۶]

در این جبر، اجزای سیستم به صورت پروسه های انتزاعی نمایش داده می شوند که با یکدیگر ارتباط دارند. پروسه ها به صورت جعبه سیاه دیده می شوند که تنها رفتار خارجی آنها مورد بررسی قرار می گیرد. همچنین پروسه ها به کمک مکانیزم **interaction point** با یکدیگر رابطه برقرار می کنند. در LOTOS به این نقاط **event gate** یا به اختصار **gate** گفته می شود (شکل ۲,۳).



شکل ۲,۳. نقطه **gate** در جبر پروسه ها [۴]

Event نشان دهنده یک همگامی^۱ بین دو یا چند پروسه است. به طور کلی سه نوع **event**

تعریف شده اند:

- **Pure synchronization**: هیچ مقداری بین پروسه ها تبادل نمی شود.
- **Value establishment**: مقادیر ارائه شده توسط یک یا چند پروسه از سوی بقیه مورد پذیرش نیست.
- **Value negotiation**: مقادیر از سوی یک یا چند پروسه مورد قبول است.

همچنین یک عبارت رفتاری^۲ انتخابی از بین مجموعه ای از **event** ها است که برای یک محیط مطرح می گردد. در شرایطی که امکان همگام سازی بین **event** های پیشنهاد شده در یک محیط

¹ Process Algebra

² Synchronization

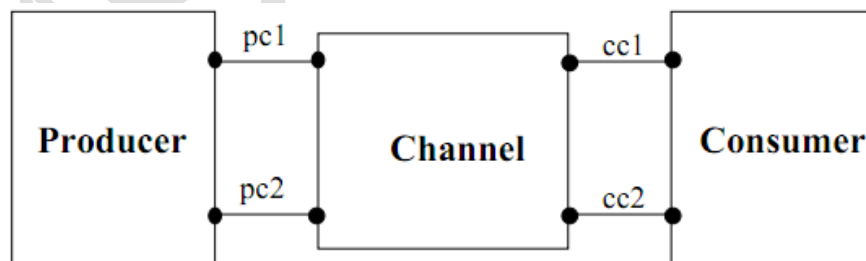
³ Behavior expression

وجود نداشته باشد یک بن بست^۱ رخ داده است. همچنین اگر هیچ پیشنهادی در محیط مطرح نشده باشد یک livelock رخ می دهد (مانند یک حلقه بینهایت).

همچنین LOTOS مفهوم ترتیب زمانی eventها را نیز به قواعد جبری موجود اضافه نموده است. این زبان از عملگرهای temporal برای ترکیب عبارات رفتاری و به دست آوردن عبارات رفتاری پیچیده تر استفاده می کند. در عین حال این عملگرها از قواعد خوش تعریف پیروی می نمایند که تفسیر صریح و یکسانی را برای افراد ممکن می سازد.

به عنوان خلاصه ای از خصوصیات LOTOS، می توان آنرا تعریف فرمالی از ترکیب داده و کنترل، بر مبنای جبر پروسه ها دانست که همزمانی^۲ یکی خصوصیات ذاتی آن است. همچنین قواعد LOTOS به دلیل عملیاتی بودن قابل تفسیر و اجرا^۳ می باشد و ماژولار است [۱۷].

در ادامه مثالی از مدلسازی مساله تولید کننده- مصرف کننده^۴ با این زبان آورده شده است. در این مساله سه ماژول تولید کننده، مصرف کننده و کانال که وظیفه هماهنگ سازی این دو را به عهده دارد حضور دارند. لازم به ذکر است که تغییراتی در صورت مساله به وجود آمده تا خصوصیات LOTOS بهتر نشان داده شود. از آن جمله، تولید کننده می تواند دو خروجی را به صورت همزمان یا غیر همزمان تولید نماید و سپس متوقف شود. همچنین مصرف کننده پس از مصرف ترتیبی یا همزمان آنها متوقف می گردد (شکل ۲،۴).



شکل ۲،۴. مدل تولید کننده- مصرف کننده به کمک LOTOS [۱۷]

¹ Deadlock

² Concurrency

³ Execution

⁴ Producer- Consumer

برای کنترل پیچیدگی مساله، در ابتدا عمل تجزیه^۱ را روی این سه بخش انجام داده و هر یک را به عنوان یک پروسه مستقل تشریح می کنیم. سپس آنرا را با هم ترکیب می نمائیم.

با توجه به تعاریف pc1 و pc2 به عنوان کانال های تولید کننده و cc1 و cc2 به عنوان کانال های مصرف کننده و همچنین عملگرهای انتخاب ([]) و فعال سازی (>>) می توان پروسه های فوق را به صورت زیر تعریف نمود:

پروسه تولید کننده:

```
process Producer [pc1, pc2] : exit :=
pc1; pc2; exit
endproc
```

پروسه کانال:

```
process Channel [ pc1, pc2, cc1, cc2] : exit :=
pc1;
(
pc2; cc1; exit
[]
cc1; pc2; exit
)
>> cc2; exit
endproc
```

پروسه مصرف کننده:

```
process Consumer [cc1, cc2] : exit :=
cc1; cc2; exit
endproc
```

¹ Decomposition

² Enable

در این بین می توان جزئیات اجرایی بیشتر مانند عملکرد داخلی هریک از پروسه ها را با عملگرهای بیشتری مدل نمود به عنوان مثال مساله گم شدن یکی از ورودی ها به کانال به شکل زیر قابل توصیف است:

```
process Channel [pc1, pc2, cc1, cc2] : exit :=
pc1; ( pc2 ; cc1; exit [] cc1; pc2; exit [] i; pc2; exit )
>> ( cc2; exit [] i; exit)
Endproc
```

در نهایت با تعریف عملگر میانه گزاری^۱ (|||) برای شبیه سازی توازی^۲ بین پروسه ها می توان ترکیب سه پروسه یاد شده را به شکل زیر نوشت:

```
1  specification Producer_Consumer [ pc1, pc2, cc1 cc2 ] : exit
2
3  behavior
4  (
5  Producer [ pc1, pc2 ]
6  |||
7  Consumer [ cc1, cc2 ]
8  )
9  ||
10 Channel [pc1, pc2, cc1, cc2]
11
12 where
13 process Producer [ out1, out2 ] : exit := ... (*As defined
    previously*)
14 process Consumer [ in1, in2 ] : exit := ... (*As defined
    previously *)
```

¹ Interleaving operator

² Parallelism

15 process Channel [le1, le2, , re1, re2] : exit := ... (*As defined
previously *)
16 endspec

برای جزئیات بیشتر در مورد مثال فوق و عملگرهای زبان LOTOS به [۱۷] و برای مطالعه بیشتر در مورد زبان LOTOS به [۳۳] الی [۳۴] مراجعه کنید.

۲.۳. VDM-SL

VDM-SL^۱ ابزاری برای توصیف مسائل نرم افزاری است که به صورت مجرد و مستقل از جزئیات ماشین ابداع شده است. با توجه به اینکه VDM یک زبان بوده و دارای semantic فرمال می باشد، ابزاری برای ارزیابی آن نیز ارایه شده است که می توان الگوریتم های مدل شده در این زبان را ارزیابی نمود. همچنین ابزاری برای اجرای سیستم مدل شده و مشاهده نتیجه تست برای آن پیش بینی شده تا کسانی که با VDM آشنایی ندارند بتوانند سیستم مورد نظر خود را بررسی کنند. با این حال این زبان قابلیت مدل سازی سیستم های موازی را ندارد.

با توجه به اینکه VDM یک زبان model-based است به طور کلی تفاوت بین این گروه از ابزار فرمال و متدهای جبری را می توان اینگونه توضیح داد که در گروه اول، ما عملیات در یک سیستم را بر اساس اینکه چه تأثیری بر مقادیر می گزارند توصیف می کنیم. در مقابل در ابزار جبری، خصوصیات عملیات را به صورت مجرد شرح می دهیم [۱۸].

با توجه به اینکه بیشترین توان VDM در مدل سازی سیستم های خطی است، به نظر می رسد این زبان بیشتر به توصیف خصوصیات کارکردی سیستم می پردازد. بنابراین بخش بزرگی از توصیف به بیان انواع داده ای ساده و مجرد و نیز ADT ها مربوط است. در چنین حالتی بسیاری از ساختارهای عبارات جبری قابل نگاشت به این زبان هستند.

¹ Vienna Development Method- Specification Language

در ادامه مدل ساده‌ای از VDM برای مدل سازی حساب بانکی آورده شده است [۱۸]:

در این مثال مشتری با CustNum و حساب با AccNum مدل می شود.

```
AccNum = token;  
CustNum = token;  
Balance = int;  
Overdraft = nat;  
AccData :: owner : CustNum  
          balance : Balance  
state Bank of  
  accountMap : map AccNum to AccData  
  overdraftMap : map CustNum to Overdraft  
  inv mk_Bank(accountMap,overdraftMap) == for all a in set rng  
accountMap & a.owner in set dom overdraftMap and  
a.balance >= -overdraftMap(a.owner)
```

برای مطالعه بیشتر به [۳۵] الی [۳۶] مراجعه کنید.

۲،۴. شبکه های پتری

شبکه های پتری^۱ یکی از قدیمیترین زبان های فرمال برای توصیف و ارزیابی سیستم ها است. از بارزترین خصوصیات این زبان می توان به توانایی آن برای توصیف سیستم های موازی اشاره نمود. همچنین به دلیل پختگی^۲، این زبان مکانیزم هایی برای بیان جنبه های کارکردی و رفتاری سیستم

¹ Petri nets

² Maturity

ارائه می نماید. همچنین علاوه بر توانایی توصیف سیستم ها، می تواند شیوه هایی برای تحلیل^۱ و ارزیابی^۲ و نیز در نهایت اثبات فرمال آنها^۳ ارائه نماید [۱۹] الی [۲۲].
بر اساس تعریف، یک شبکه پتری به صورت زیر به شکل فرمال تعریف می شود [۲۳]:

$$P = \{p_1, p_2, \dots, p_m\}$$

مجموعه متناهی از موقعیت ها^۴

$$T = \{t_1, t_2, \dots, t_n\}$$

مجموعه متناهی از گزارها^۵

$$F \subseteq (P \times T) \cup (T \times P)$$

مجموعه ای از لبه های گراف^۶ (روابط بین p ها و t ها)

$$W: \rightarrow \{1, 2, 3, \dots\}$$

تابع وزن گذاری روابط

$$M_0: P \rightarrow \{0, 1, 2, \dots\}$$

وضعیت اولیه سیستم.

همچنین ضروری است :

$$P \cap T = \emptyset \text{ and } P \cup T \neq \emptyset$$

به عنوان مثال مدل ساده ای از یک پروتکل ارتباطی در شکل ۲,۵ نشان داده شده است.

¹ Analysis

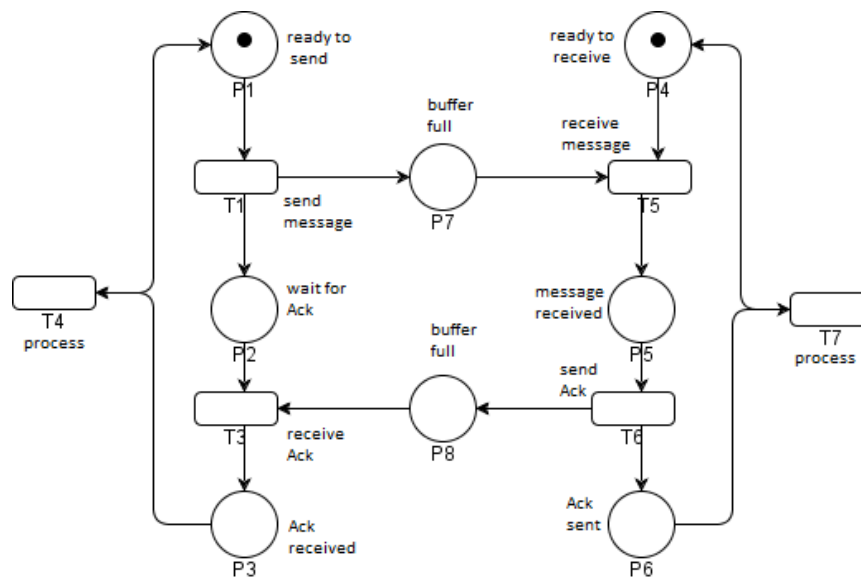
² Validation

³ Verification

⁴ Places

⁵ Transitions

⁶ Arcs



شکل ۲،۵. مثالی از مدل سازی یک پروتکل به کمک شبکه های پتری [۲۳]

در شکل بالا، وضعیت شروع $M_0 = (1,0,0,1,0,0,0,0)$ می باشد که به معنی فعال بودن موقعیت های p_1 و p_4 است. بنابراین گزاره های t_1 و t_5 به ترتیب فعال هستند. باید توجه داشت که فعال بودن یک گزار لزوماً به معنی اجرا شدن آن نیست بلکه اجرا شدن به رخ دادن اتفاق در سیستم واقعی بر می گردد. همچنین اجرا شدن t_1 و t_5 به ترتیب توکن های p_1 و p_4 را حذف کرده و توکن هایی به p_2 و p_5 اضافه می کند.

۲،۴،۱. خصوصیات رفتاری^۳

با تعریف وضعیت شروع^۴ به عنوان وضعیت اولیه سیستم ، خصوصیات رفتاری در واقع جنبه هایی از سیستم هستند که به این وضعیت شروع وابسته هستند و در موقعیت های اجرایی سیستم تغییر می کنند.

¹ Enable

² Firing

³ Behavioral properties

⁴ Initial marking

در شبکه (N, M_0) (شبکه پتری N با وضعیت شروع M_0)، مجموعه تمام دنباله های گزارهای قابل اجرا که از وضعیت M_0 شروع می شوند را با $L(N, M_0)$ و یا به شکل ساده تر با $L(M_0)$ ^۱ نشان می دهند.

به کمک پارامترهای زیر می توان جنبه های رفتاری سیستم توصیف نمود [۲۳] و [۲۴]:

۲.۴.۱.۱. Reachability

طبق تعریف، وضعیت سیستم در M_n ، از وضعیت M_0 ، reachable خواهد بود اگر دنباله ای از اجرای گزارها وجود داشته باشد که M_0 را به M_n برسد. مجموعه وضعیت های قابل دسترسی از طریق M_0 را با مجموعه $R(M_0)$ نشان می دهیم.

۲.۴.۱.۲. Boundedness

شبکه پتری (N, M_0) ، k -bounded یا به اختصار bounded خواهد بود اگر تعداد توکن های موجود در هر place، برای همه وضعیت های قابل دستیابی از وضعیت شروع، از تعداد محدود k تجاوز نکند. به عبارت دیگر $M(p) \leq k$. همچنین یک شبکه پتری 1-bounded، شبکه پتری ایمن^۳ نامیده می شود.

۲.۴.۱.۳. Liveness

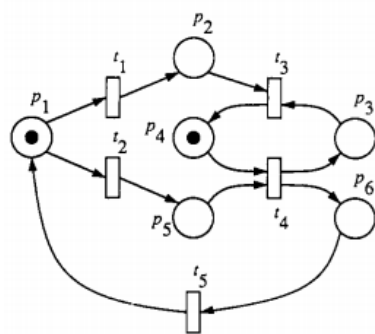
این مفهوم در ارتباط نزدیک با مفهوم عدم وجود بن بست در سیستم عامل ها می باشد. شبکه پتری (N, M_0) ، live خوانده می شود (به عبارت دیگر وضعیت M_0 یک وضعیت live برای N شمرده می شود) اگر، بدون توجه به اینکه چه وضعیتی از M_0 پیش می آید، در نهایت اجرای تمام گزارهای شبکه با دنبال کردن دنباله ای از گزارها ممکن باشد.

^۱ Language (M_0)

^۲ Firing sequence

^۳ Safe

این بدان معنی است که یک شبکه پتری live عملیات بدون بن بست را تضمین می کند بدون اینکه ترتیب اجرای گزارها مهم باشد. به عنوان نمونه، شبکه شکل ۲,۶ دارای خصوصیت Liveness نمی باشد زیرا در صورتی که گزار t_1 در ابتدا اجرا شود، دیگر هیچ یک از گزارها امکان اجرا شدن نخواهند داشت.



شکل ۲,۶. نمونه ای از شبکه پتری non-Live [۲۳]

Liveness خصوصیتی ایده آل برای بسیاری از سیستم ها است. با این حال اثبات و فراهم کردن کامل این خصوصیت دشوار برای بسیاری از سیستم های پیچیده غیر ممکن و هزینه بر است. سیستم عامل کامپیوترهای بزرگ نمونه ای از چنین سیستم هایی است. بنابراین در عمل این خصوصیت نادیده گرفته شده و چندین سطح از Liveness برای شبکه ها در نظر گرفته می شود. بر این اساس سطح Liveness گزار t در شبکه پتری (N, M_0) یکی از موارد زیر خواهد بود:

0) Dead (L0-live)

اگر t هیچ گاه و در هیچ یک از دنباله های اجرای $L(M_0)$ اجرا نشود.

1) L1-live (probably fireable)

اگر t حداقل یک بار و در یکی از دنباله های اجرای $L(M_0)$ بتواند اجرا شود.

2) L2-live

اگر با در نظر گرفتن عدد مثبت k ، گزار t بتواند حداقل k بار در دنباله های اجرای مختلف $L(M_0)$ اجرا شود.

3) L3-live

اگر t به دفعات نامحدودی در تعدادی از دنباله های اجرای $L(M_0)$ ظاهر شود.

4) L4-live (Live)

اگر t در هر وضعیت M در $R(M_0)$ در مرتبه $L1$ -live باشد.

■

گفته می شود شبکه پتری (N, M_0) Lk -live است اگر همه گزاره های موجود در شبکه حداقل از مرتبه Lk -live باشند و $k=0,1,2,3,4$. همچنین $L4$ -liveness قویترین سطح $live$ بودن سیستم و و معادل مفهومی است که در ابتدای بخش توضیح داده شد.

به سادگی می توان دریافت که یک گزار یا شبکه که در سطح k $live$ است، لزوما در سطح $k-1$ نیز $live$ خواهد بود. به عبارت دیگر رابطه دلالت بین آنها به این شکل برقرار است:

$$L4\text{-liveness} \Rightarrow L3\text{-liveness} \Rightarrow L2\text{-liveness} \Rightarrow L1\text{-liveness}$$

گفته می شود که یک گزار دقیقا Lk -live است اگر این گزار Lk -live باشد ولی $L(k+1)$ -live نباشد، $K=1,2,3$.

Reversibility ۲,۴,۱,۴

شبکه (N, M_0) $reversible$ است اگر برای هر وضعیت M ، M_0 از طریق M قابل دستیابی (reachable) باشد.

Coverability ۲,۴,۱,۵

در شبکه (N, M_0) گفته می شود وضعیت M قابل پوشش است اگر وضعیت M' در $R(M_0)$ وجود داشته باشد به طوری که $M'(p) \geq M(p)$.

- Home state

وضعیت M' در شبکه (N, M_0) Home state نامیده می شود اگر برای هر وضعیت M در $R(M_0)$ ، M' از M قابل دسترسی باشد.

۲.۴.۲. زیر مجموعه های شبکه های پتری

تعریف: یک شبکه پتری، شبکه اولیه^۱ نامیده می شود در صورتی که وزن تمام لبه های آن ۱ باشد.

با این تعریف در این بخش شبکه های اولیه ای را معرفی می کنیم که دارای خواص مشترکی هستند. البته باید توجه داشت که شبکه های اولیه و غیر اولیه هر دو دارای قدرت مدلسازی یکسانی هستند و تفاوت آنها در راحتی و سرعت استخراج و اثبات خصوصیات است. در این بخش از نشانه گذاری زیر به عنوان مقدمه استفاده می کنیم. در اینجا F مجموعه همه لبه های شبکه پتری است:

$$\bullet t = \{p \mid (p, t) \in F\}$$

مجموعه موقعیت های ورودی به t

$$t^\bullet = \{p \mid (t, p) \in F\}$$

مجموعه موقعیت های خروجی از t

$$\bullet p = \{t \mid (t, p) \in F\}$$

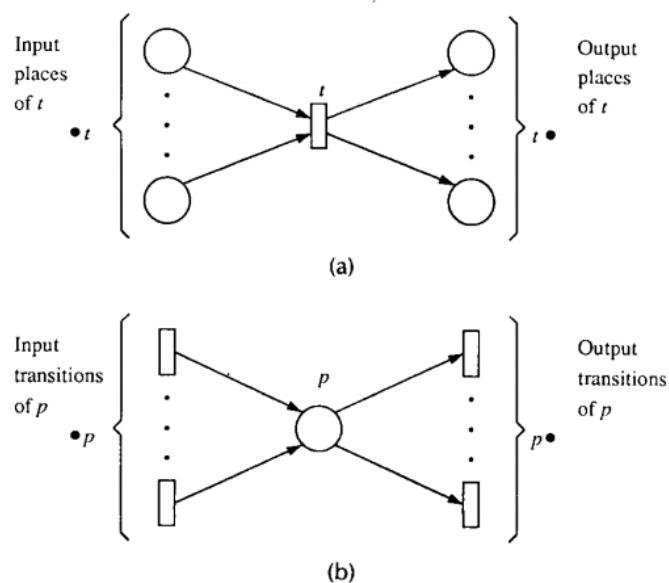
مجموعه گزارهای ورودی به p

$$p^\bullet = \{t \mid (p, t) \in F\}$$

مجموعه گزارهای خروجی از p

مثال هایی از نشانه های گفته شده در شکل ۲،۷ نشان داده شده است. با این تعریف می توانیم زیر مجموعه های شبکه های پتری را با اعمال قوانینی در ساختار آنها مشخص کنیم. در این تحقیق فرض شده است که در هیچ یک از شبکه های مورد بحث، موقعیت یا گزار ایزوله وجود ندارد یعنی گزار یا موقعیتی با شرایط $\bullet p = p^\bullet = \emptyset$ یا $\bullet t = t^\bullet = \emptyset$ وجود ندارد.

¹ Ordinary Petri net



شکل ۲،۷. نشانه گذاری برای: (a) مجموعه موقعیت های ورودی و خروجی برای t و (b) مجموعه

گزارهای ورودی و خروجی برای p [۲۳]

با مقدمه گفته شده، کلاس های زیر را برای شبکه های پتری داریم:

۲،۴،۲،۱ State Machine (SM)

یک شبکه پتری اولیه با این شرط که هر گزار t دقیقاً یک موقعیت ورودی و یک موقعیت خروجی داشته باشد. یعنی:

$$|\bullet t| = |t \bullet| = 1 \quad \text{for all } t \in T.$$

۲،۴،۲،۲ Marked Graph (MG)

یک شبکه پتری اولیه با این شرط که هر موقعیت p دقیقاً یک گزار ورودی و یک گزار خروجی داشته باشد. یعنی:

$$|\bullet p| = |p \bullet| = 1 \quad \text{for all } p \in P.$$

۲.۴.۲.۳ Free-choice net (FC)

یک شبکه پتری اولیه است که در آن هر لبه متصل به یک موقعیت، یا یک لبه خروجی یکتا از یک گزار و یا یک لبه ورودی یکتا به یک گزار است. یعنی:

$$\text{For all } p \in P, |p \bullet| \leq 1 \text{ or } \bullet(p \bullet) = \{p\}$$

که معادل است با:

$$\text{For all } p_1, p_2 \in P, p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow |p_1 \bullet| = |p_2 \bullet| = 1$$

۲.۴.۲.۴ Extended Free-choice net (EFC)

یک شبکه پتری اولیه است که:

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet = p_2 \bullet \text{ for all } p_1, p_2 \in P$$

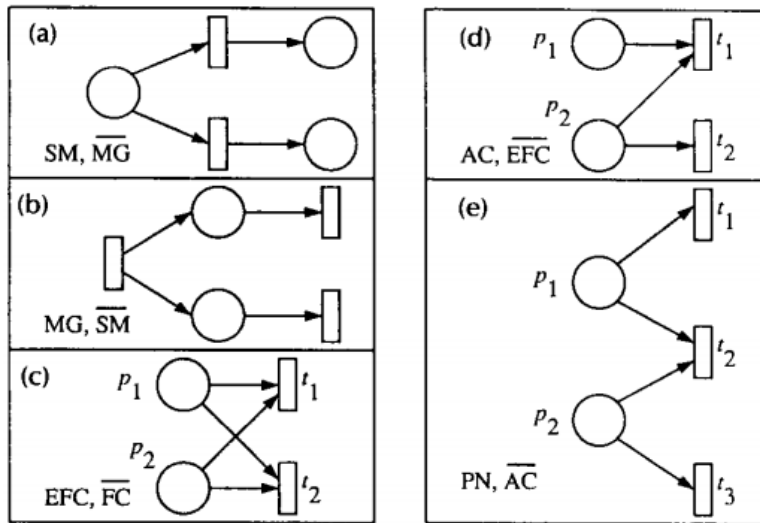
۲.۴.۲.۵ Asymmetric choice net (AC)

این کلاس که به عنوان شبکه ساده هم شناخته می شود، یک شبکه پتری اولیه است که :

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet \subseteq p_2 \bullet \text{ or } p_2 \bullet \subseteq p_1 \bullet \text{ for all } p_1, p_2 \in P.$$

مثال هایی از کلاس های تشریح شده که تفاوت های کلیدی این زیر مجموعه های شبکه پتری را نشان می دهد در شکل ۲.۸ آمده است.

¹ Simple net



شکل ۲،۸. مثال هایی از زیر مجموعه های شبکه های پتری [۲۳]

۲،۴،۳. قضایا و فرضیات [۲۳]

با یادآوری این نکته که شبکه اولیه به شبکه ای گفته می شود که وزن همه لبه های آن ۱ باشد، قضیه های زیر را داریم:

قضیه ۱: اگر شبکه پتری عادی (N, M_0) ، Live و Safe باشد، نباید در آن موقعیت source یا sink و نیز گزار source یا sink وجود داشته باشد یعنی: $x \in P \cup T, \bullet x \neq \emptyset \neq x \bullet$.

□

با بسط دادن این قضیه می توان گفت اگر شبکه پتری متصل (N, M_0) ، live و safe باشد، آنگاه شبکه N یک شبکه متصل قوی است.

قضیه ۲: اگر درخت پوشای شبکه پتری (N, M_0) را به عنوان T در نظر بگیریم، شبکه (N, M_0) ، Safe خواهد بود اگر و تنها اگر فقط مقادیر ۰ و ۱ در برچسب لبه های T وجود داشته باشد.

□

¹ Strongly connected net

قضیه ۳: اگر درخت پوشای شبکه پتری (N, M_0) را به عنوان T در نظر بگیریم، گزار t یک بن بست است اگر و تنها اگر t به عنوان برچسب یک لبه در T ظاهر نشود.

□

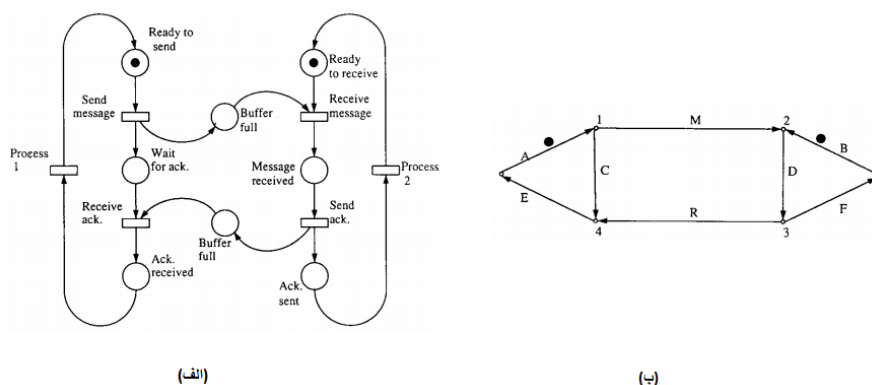
قضیه ۴: یک state machine (N, M_0) ، live است اگر و تنها اگر این شبکه متصل قوی بوده و M_0 حداقل یک توکن داشته باشد.

□

قضیه ۵: یک state machine (N, M_0) ، safe است اگر و تنها اگر M_0 حداکثر یک توکن داشته باشد. همچنین شرط لازم و کافی برای اینکه یک live state machine (N, M_0) ، safe نیز باشد این است که M_0 دقیقاً یک توکن داشته باشد.

□

در شبکه های marked graph، هر موقعیت دقیقاً یک لبه ورودی و یک لبه خروجی با وزن واحد دارد. بنابراین گراف نشانه دار (N, M_0) را می توان با گراف نشانه دار جهت دار $(G, M_0)^2$ نشان داد که در آن لبه ها معادل موقعیت ها و گره ها معادل گزارها هستند. شکل ۲،۹ مثالی از مدل پتری یک پروتکل ارتباطی در شبکه های کامپیوتری (الف) و گراف نشانه دار متناظر با آن (ب) را نشان می دهد.



شکل ۲،۹. یک شبکه پتری و گراف نشانه دار مربوط به آن [۲۳]

¹ Dead transition

² Directed marked graph

اجرای یک گره (گزار) در یک marked graph شامل حذف شدن یک توکن از از لبه های ورودی (موقعیت های ورودی) و اضافه شدن یک توکن به لبه های خروجی (موقعیت های خروجی) می باشد. اگر یک گره بر روی یک مدار جهت دار^۱ یا حلقه قرار داشته باشد، آنگاه دقیقا یکی از لبه های ورودی و یکی از لبه های خروجی آن متعلق به حلقه مذکور خواهد بود. متقابلا، اگر گره ای بر روی یک حلقه قرار نداشته باشد، هیچ یک از لبه های متصل به آن متعلق به مدار جهت دار مذکور نخواهد بود.

با توجه به مقدمه گفته شده، قضایای زیر را در مورد خصوصیات نامتغیر توکن ها در marked graph داریم:

قضیه ۶: در یک marked graph، تعداد توکن ها در یک مدار جهت دار تحت اجرای هر گزار، نا متغیر است؛ یعنی برای هر مدار جهت دار C و هر وضعیت M در $R(M_0)$ داریم:

$$M(C) = M_0(C)$$

که در اینجا $M(C)$ نشان دهنده مجموع تعداد توکن ها در C است.

□

بر اساس قضیه ۶، اگر در وضعیت اولیه شبکه، توکنی در یک مدار جهت دار وجود نداشته باشد، این حلقه بدون توکن باقی خواهد ماند. بنابراین هیچ یک از گره ها (گزار) در آن امکان اجرا شدن نخواهند داشت. از سوی دیگر اگر یک گره هیچ گاه طی هیچ وضعیتی امکان اجرا شدن نداشته باشد (L0-live transition)، می توان با بررسی معکوس دنباله گزارها، مدار مستقیم بدون توکن را پیدا کرد. بر این اساس قضیه ۷ را داریم:

قضیه ۷: گراف نشانه دار (G, M_0) ، live است اگر و تنها اگر در وضعیت M_0 ، حداقل یک توکن در هریک از مدارهای جهت دار G وجود داشته باشد.

□

قضیه ۸: گراف نشانه دار (G, M_0) live، safe خواهد بود اگر و تنها اگر هر لبه ای (موقعیت)

در این گراف متعلق به مدار جهت دار C باشد به طوری که $M_0(C) = 1$.

¹ Directed circuit

² Invariant property

□

لم ۱: با استفاده از گراف پوشای مدل پتری (N, M_0) تعدادی از خصوصیات آن را می توان بررسی نمود. از آن جمله می توان به موارد زیر اشاره کرد:

- ۱- شبکه (N, M_0) ، bounded و در نتیجه $R(M_0)$ متناهی است اگر و تنها اگر مقدار ω (تعداد بی نهایت توکن در یک موقعیت) در هیچ یک از برچسب های گره های گراف وجود نداشته باشد.
- ۲- شبکه (N, M_0) ، safe است اگر و تنها اگر فقط مقادیر ۰ و ۱ در برچسب گره های گراف دیده شود.
- ۳- گزار t در شبکه (N, M_0) ، dead خواهد بود اگر و تنها اگر این گزار به صورت برچسب یک لبه در گراف پوشا ظاهر نشود.
- ۴- اگر وضعیت M از طریق M_0 قابل دسترسی (reachable) باشد. آنگاه در گراف گره ای با برچسب M' وجود دارد به طوری که $M \leq M'$

□

برای مطالعه بیشتر قضایا و فرضیات به [۲۵] الی [۲۸] مراجعه کنید.

۲.۵. جمع بندی

در فصل دو، با تکیه بر مطالعات انجام شده در زمینه ابزار توصیف فرمال سیستم ها، چهار ابزار مختلف را انتخاب و بررسی نمودیم. هر یک از این چهار ابزار، خواه مبنی بر ابزار ریاضی مانند گراف باشند و یا مبتنی بر روابط جبری، به طراح در به دست آوردن تصویری قابل درک از سیستم واقعی کمک می کنند. این تصویر می تواند با متد های قابل دفاع مورد بررسی و تحلیل قرار گرفته و ظرفیت ها و مشکلات سیستم را از جنبه های زیادی به طراح نشان دهد.

با توجه به تنوع سیستم های دنیای واقعی و خصوصیات بی شمار آنها، به طبع، ابزار فرمال متنوعی نیز برای بررسی آنها ابداع شده است. به عنوان نمونه بسیاری از ابزار مدلسازی فرمال قادر به نشان دادن دقیق خصوصیات سیستم های غیر موازی هستند. در مقابل بسیاری می توانند خصوصیات

سیستم های موازی را که بسیار پیچیده ترند نشان دهند. همچنین خصوصیات ساختاری و رفتاری سیستم ها و نحوه برخورد آنها با زمان مسائل پیچیده ای است که باید در مدل سازی سیستم با ابزار فرمال مد نظر داشت.

با توجه به توضیحات ارائه شده در بخش های گذشته، می توان جدول مقایسه ای برای چهار ابزار توصیف فرمال مذکور به صورت جدول ۲،۱ ارائه داد.

جدول ۲،۱. خلاصه مقایسه ابزار توصیف فرمال

ردیف	خصوصیت	ابزار توصیف فرمال			
		Petri nets	VDM-SL	LOTOS	ASM
۱	پارادایم	مبتنی بر مدل	مبتنی بر مدل	مبتنی بر جبر	مبتنی بر جبر
۲	سطح فرمالیتی	فرمال	فرمال	فرمال	فرمال
۳	ابزار گرافیکی برای نمایش	دارد	ندارد	ندارد	ندارد
۴	قابلیت توصیف سیستمهای موازی	دارد	ندارد	دارد	ندارد
۵	قابلیت بیان ^۱	قوی	ضعیف	ضعیف	ضعیف
۶	قابلیت انتزاع پذیری	قوی	متوسط	متوسط	متوسط
۷	پشتیبانی از شئی گرایی	دارد	ندارد	دارد	ندارد
۸	قابلیت تبدیل به کد اجرایی	دارد	دارد	دارد	دارد
۹	استفاده از متغیرها	بله	بله	بله	بله
۱۰	قابلیت توصیف سیستم های غیر تعینی ^۲	دارد	ندارد	دارد	ندارد
۱۱	قابلیت اثبات پذیری	دارد	دارد	دارد	دارد

^۱ Expressibility

^۲ Non deterministic

۱۲	امکان بررسی مدل ^۱	ندارد	ندارد	دارد	دارد
----	------------------------------	-------	-------	------	------

با توجه به نتایج به دست آمده از مقایسه ابزار مذکور و ماهیت سیستم مورد بررسی، در این تحقیق از زبان شبکه های پتری برای مدل سازی فرمال سیستم استفاده خواهد شد. به همین دلیل در این فصل در مورد این ابزار تمرکز و بررسی بیشتری به کار رفته است. همچنین در پایان کلیه قضایا و فرضیاتی که در طول تحقیق از آنها بهره گرفته می شود آمده است.

در فصل بعد به تشریح تکنولوژی مجازی سازی خواهیم پرداخت. این تکنولوژی برای ساخت لایه نرم افزاری دیتا سنتر به کار خواهد رفت. در نهایت در فصل چهار، مدل های فرمالی در چند لایه از دیتا سنتری که به کمک این تکنولوژی ساخته شده طراحی خواهیم کرد.

¹ Model chcking

فصل سوم: بررسی معماری دیتا سنترها

در این بخش لازم است معماری زیر ساخت نرم افزاری دیتا سنترها را بررسی و تحلیل نمائیم. این تحلیل پیش نیاز ما در طراحی مدل فرمال این زیر ساخت خواهد بود. به این منظور، در ابتدا معماری نرم افزارهای مدیریت ماشین های مجازی تحت عنوان نرم افزارهای Hypervisor را بررسی و تشریح می نمائیم [۳۷] و [۳۸].

در حال حاضر سه نرم افزار مطرح در این زمینه، بیشترین سهم بازار سیستم های مجازی سازی را در اختیار دارند. این سه محصول عبارتند از:

- Microsoft Hyper-V
- Xen
- VMware ESXi

از دیدگاه های مختلفی می توان hypervisor ها را با یکدیگر مقایسه نمود. از آن جمله می توان به پیاده سازی سرویس مجازی سازی کامل^۱ در مقابل مجازی سازی ضمنی^۲ اشاره کرد. در حالت اول، سخت افزار به طور کامل شبیه سازی می گردد و انتزاع کاملی از آن برای ماشین های مجازی ایجاد می گردد. به این ترتیب سیستم های عامل میهمان^۳ نیازی به اصلاح برای کار بر روی این ماشین مجازی نخواهند داشت. همچنین آنها و نرم افزارهای در حال اجرا بر روی آنها هیچ اطلاعی از وجود ماشین مجازی نخواهند داشت. در نتیجه پیاده سازی به شکل کامل امکان مهاجرت

¹ Full virtualization

² Para-virtualization

³ Guest Operating System

نرم افزارها و سیستم های عامل میهمان بین ماشین های فیزیکی را میسر می سازد. همچنین با ایزوله کردن نرم افزارها و سیستم عامل ها، امنیت هر یک از آنها در حد بالایی تامین خواهد شد. در عین حال این شکل از مجازی سازی مشکلات کاهش کارایی را نیز به دنبال دارد. زیرا hypervisor مجبور است تصویر کاملی از هر ماشین مجازی و سیستم عامل میهمان نصب شده بر روی آن را نگه دارد. به عنوان مثال حتی BIOS شبیه سازی می گردد. بنابراین منابع بیشتری صرف مدیریت و اجرای hypervisor خواهد شد.

در مقابل، پیازی سازی مجازی سازی به شکل ضمنی اجازه دسترسی سیستم های عامل میهمان را به بخش هایی از منابع بدون شبیه سازی آنها فراهم می کند. در این حالت لازم است روی سیستم های عامل میهمان اصلاحاتی صورت گیرد تا امکان برقراری ارتباط با hypervisor را داشته باشند. در نتیجه آنها از وجود ماشین مجازی مطلع خواهند بود. این روش در کنار برخی از مشکلات امنیتی، عموماً از سیستم های عامل closed-source پشتیبانی نمی کند زیرا امکان انجام اصلاحات لازم برای کار در ماشین مجازی در مورد آنها وجود ندارد [۳۹].

نرم افزارهای Hyper-V و Xen به صورت مجازی سازی ضمنی و ESX به صورت مجازی سازی مطلق پیاده سازی شده اند.

در ادامه به بررسی معماری هر یک از این سه محصول می پردازیم و در نهایت برای مدل سازی روابط بر روی VMware ESX متمرکز خواهیم شد.

۳.۱. Microsoft Hyper-V

توضیحات زیر از سایت رسمی اسناد فنی شرکت میکروسافت (MSDN) برداشت شده است [۴۰].

Hyper-V یک تکنولوژی مجازی سازی مبتنی بر Hypervisor است که برای Windows server 2008 نسخه ۶۴ بیتی ساخته شده است.

اولین مفهوم مطرح در این تکنولوژی نوعی از جدا سازی¹ به نام پارتیشن است. در واقع Hyper-V از این بخش های جدا شده برای نگهداری اطلاعات و اجرای سیستم عامل های مختلف استفاده می کند. وجود حداقل یک پارتیشن ضروری است که پارتیشن ریشه یا والد نامیده می شود. در این پارتیشن یک سیستم عامل Windows server 2008 نصب و اجرا می گردد که وظیفه مدیریت دیگر سیستم عامل های مهمان و برقراری ارتباط آنها با سخت افزارهای فیزیکی را بر عهده دارد. پارتیشن ریشه تنها پارتیشنی است که دسترسی مستقیم به سخت افزار فیزیکی را دارد و از طریق رابط برنامه نویسی³ hypercall سیستم عامل های میهمان را ایجاد می کند.

سیستم عامل های میهمان با تصویری از پردازنده و وقفه ها⁴ و در فضای محدودی از حافظه کار می کنند. در واقع hypervisor وقفه ها را برای سیستم عامل ترجمه کرده و به پارتیشن متناظر ارسال می کند. همچنین سیستم عامل ها (میهمان) به کمک ماژول VDev⁵ با تصویری از سخت افزارهای جانبی کار می کنند که توسط hypervisor اجرا می شود. درخواست سیستم عامل از سخت افزار مجازی توسط VMBus⁶ که یک کانال ارتباطی منطقی بین پارتیشن ها است به hypervisor و یا پارتیشن ریشه منتقل می شود و از آنجا به سخت افزار واقعی انتقال می یابد. به عبارت دقیقتر، ماژول VSC⁸ بر روی سیستم عامل میهمان درخواست خود را از طریق VMBus برای ماژول VSP⁹ بر روی سیستم عامل میزبان در پارتیشن ریشه ارسال می نماید.

همچنین می توان از امکان Enlightened I/O برای افزایش سرعت و کارایی دستیابی به سخت افزارهای جانبی مانند دیسک، رابط شبکه و غیره استفاده نمود. در واقع این تکنولوژی، پیاده سازی پروتکل های سطح بالا (مانند SCSI) است که برای استفاده در محیط مجازی آماده پیاده سازی پروتکل های سطح بالا (مانند SCSI) است که برای استفاده در محیط مجازی آماده شده اند. این

¹ Isolation

² Root (parent)

³ API

⁴ Interrupts

⁵ Virtual Devices

⁶ Device

⁷ Virtual Machine Bus

⁸ Virtualization Service Consumers

⁹ Virtualization Service Providers

¹ virtualization-aware

پروتکل ها با استفاده مستقیم از VMBus، لایه های میانی شبیه سازی سخت افزارها و نیاز به راه اندازها را از بین می برند. اما استفاده از این امکان به راه اندازی سرویس های مجتمع سازی نیاز دارد که برای چند سیستم عامل مشخص موجود است.

لیست سیستم های عاملی که توسط Hyper-V پشتیبانی می شوند در جدول ۳،۱ آورده شده است.

جدول ۳،۱. سیستم های عامل قابل پشتیبانی توسط Hyper-V R2 [۴۲]

Guest OS	Virtual processors	Edition(s)
Windows 7	1,2 or 4	Both x86-32 and x86-64, all editions except home editions
Windows Server 2008 R2	1,2 or 4	Web, Standard, Enterprise, Datacenter
Windows Server 2008	1,2 or 4	Both x86 and x64, Web, HPC, Standard, Enterprise, Datacenter, with or without Hyper-V
Linux (only including SUSE Linux Enterprise Server 10 with SP3 or version 11 and Red Hat Linux versions 5.2-5.5)	1,2 or 4	Both x86 and x64
Windows Server 2003	1 or 2	Both x86 and x64, Standard, Enterprise, Datacenter, SP2 required
Windows Server 2003 R2	1 or 2	Web, Standard, Enterprise, Datacenter, both x86 and x64 except for Web whose 64-bit version is not supported

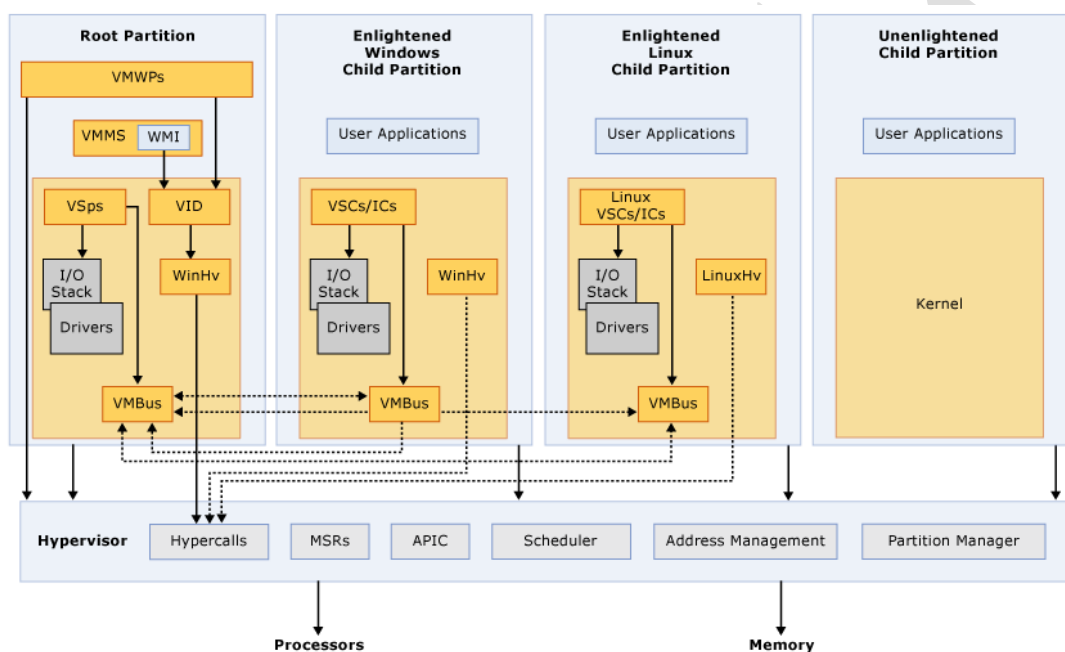
¹ Emulation

² Device drivers

³ Hyper-V integration services

Windows Vista	1 or 2	Both x86 and x64, all editions except home editions
Others	1	N/A

شمایی از معماری Hyper-V در شکل ۳،۱ آورده شده است.



شکل ۳،۱. معماری سطح بالای Hyper-V [۴۰]

۳،۱،۱. بررسی اجزاء معماری Hyper-V

در ادامه به توضیح مختصری درباره بخش ها و اصطلاحات این معماری می پردازیم.

۳،۱،۱،۱. APIC^۱

^۱ Advanced Programmable Interrupt Controller

ماژولی که امکان اولویت بندی وقفه ها را فراهم می سازد.

Child Partition ۳,۱,۱,۲

پارتیشنی که حاوی یک سیستم عامل میهمان است. تمام دسترسی های لازم به منابع فیزیکی سیستم مانند حافظه و وسایل جانبی توسط VMBus و hypervisor برای این پارتیشن فراهم می گردد.

Hypercall ۳,۱,۱,۳

رابطی برای ارتباط با hypervisor است که از طریق آن سیستم های عامل میهمان می توانند به ابزار بهینه hypervisor دست یابند.

Hypervisor ۳,۱,۱,۴

یک لایه نرم افزاری که مابین سخت افزار و یک یا چند سیستم عامل قرار می گیرد و وظیفه اصلی آن فراهم آوردن فضای مجزایی به نام پارتیشن برای اجرای هر یک از سیستم های عامل است. Hypervisor دسترسی به سخت افزار را کنترل و مدیریت می کند.

IC^۱ ۳,۱,۱,۵

ماژولی است که امکان ارتباط سیستم های عامل میهمان را با یکدیگر و با hypervisor فراهم می آورد.

I/O stack ۳,۱,۱,۶

همان پشته ورودی و خروجی است.

¹ Integration component

Root Partition ۳,۱,۱,۷

پارتیشنی است که سیستم عامل آن وظایف مدیریتی در سطح ماشین را انجام می دهد که عبارتند از راه انداز سخت افزارهای جانبی، مدیریت انرژی مصرفی و اضافه و حذف کردن دستگاه های در حال کار. این پارتیشن تنها پارتیشنی است که دسترسی مستقیمی منابع فیزیکی سیستم دارد.

VID^۱ ۳,۱,۱,۸

عملیاتی مانند سرویس های مدیریت پارتیشن ها، سرویس مدیریت پردازنده های مجازی، و سرویس مدیریت حافظه برای پارتیشن ها را ارائه می دهد.

VMBus ۳,۱,۱,۹

یک مکانیزم ارتباطی مبتنی بر کانال برای ارتباط پارتیشن ها و دستگاه ها فراهم می آورد.

VMMS^۲ ۳,۱,۱,۱۰

مدیریت وضعیت همه ماشین های مجازی در پارتیشن های فرزند را بر عهده دارد.

VMWP^۳ ۳,۱,۱,۱۱

یک بخش user mode در پشته مجازی سازی است. با این تعریف که پروسه ایجاد کننده^۴ یک سرویس مدیریت ماشین ها مجازی در پارتیشن ریشه (windows server 2008) ایجاد می کند که به کمک آن می توان سیستم های عامل میهمان را مدیریت نمود، سرویس VMWP یک پروسه ایجاد کننده مجزا برای هر یک از پارتیشن های فرزند ایجاد می کند.

^۱ Virtualization Infrastructure Driver

^۲ Virtual Machine Management Service

^۳ Virtual Machine Worker Process

^۴ Worker process

VSC^۱،۱،۱۲

یک ماژول ترکیبی که در پارتیشن فرزند قرار می گیرد و نحوه استفاده از منابع سخت افزاری را که توسط سرویس دهنده مجازی (VSP) ارائه شده بهینه می نماید. VSC با VSP متناظر خود در پارتیشن ریشه از طریق VMBus ارتباط برقرار می کند تا به درخواست های I/O پارتیشن فرزند سرویس دهد.

VSP^۲،۱،۱۳

سرویزی که در پارتیشن ریشه قرار می گیرد و از درخواست های پارتیشن های فرزند که از طریق VMBus مطرح می شود پاسخ می دهد.

WinHv^۳،۱،۱۴

این ماژول پلی بین راه اندازهای سیستمهای عامل موجود در پارتیشن ها و hypervisor است که به راه اندازها امکان می هد تا از طریق رابط فراخوانی استاندارد ویندوز، hypervisor را صدا بزنند.

WMI^۴،۱،۱۵

ابزار مدیریت ویندوز مجموعه ای از API ها هستند که سرویس مدیریت ماشین های مجازی برای مدیریت و کنترل ماشین ها ارائه می دهد.

۳،۱،۲. نقاط ضعف

¹ Virtualization Service Client

² Virtualization Service Provider

³ Windows Hypervisor Interface Library

⁴ Windows Management Instrumentation

در مجموع می توان گفت که Hyper-V زیر ساخت نسبتاً سبکی برای پیاده سازی ساختار مجازی سازی است. در عین حال به دلیل نیاز به وجود بخشی از معماری در سیستم های عامل میهمان، اجرای آنها کاملاً مستقل از hypervisor نخواهد بود. در عمل نیز می بینیم که Hyper-V از تعداد محدودی از سیستم های عامل به عنوان میهمان پشتیبانی می کند تعدادی از آنها در جدول ۳،۱ آمده است.

همچنین نیاز به وجود یک سیستم عامل میزبان^۱ هنوز هم وجود دارد. به این معنی که یک Windows server 2008 باید در پارتیشن ریشه نصب گردد تا بتواند درخواست های سیستم های عامل میهمان از سخت افزارهای مجازی را به سخت افزارهای فیزیکی منتقل نماید [۴۳]. برای مطالعه بیشتر در مورد Microsoft Hyper-V به [۴۴] الی [۴۸] مراجعه کنید.

۳،۲. Xen

در این بخش به تشریح کلیات معماری نرم افزار مجازی سازی Xen خواهیم پرداخت [۴۹] و [۵۰]. این hypervisor معروفترین نرم افزار مجازی سازی با متن باز است که شرکت ها و سازمان های متعددی با بومی سازی از آن برای پیاده سازی زیر ساخت مجازی خود استفاده کرده اند. Xen به صورت یک مجازی ساز ضمنی پیاده سازی شده است بنابراین برای سیستم های عامل میهمان برای کار با آن نیاز به یک سری تغییراتی دارند.

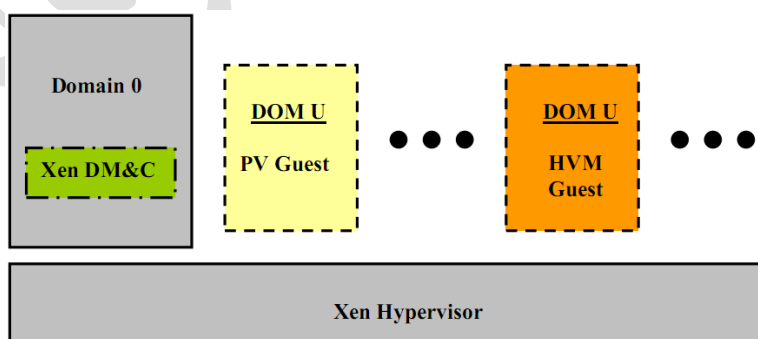
لیست سیستم های عاملی که توسط Xen 3.0 پشتیبانی می شوند در جدول ۳،۲ آورده شده است.

جدول ۳،۲. سیستم های عامل قابل پشتیبانی توسط Xen نسخه ۳

¹ Host OS

Operating System	Runs as Dom0 (host os)	Runs as DomU (guest os)
Linux 2.6	Yes	Yes
NetBSD 3.1	No	Yes
NetBSD 4.0_BETA2 and -CURRENT	Yes	Yes
FreeBSD 5.3	No	currently broken? Actively being worked on
FreeBSD 7-CURRENT	no	can be patched; works.
Plan 9	No	currently broken?
ReactOS	No	planned, development stalled
Solaris 10	Unknown	Yes
Un-Modified OS	No	Initial support for unmodified guests when using Intel VTX hardware, e.g. Windows

این hypervisor از بخش های مختلفی تشکیل شده که در ادامه به تشریح آنها می پردازیم. همچنین شمایی کلی از آن در شکل ۳,۲ آمده است.



شکل ۳,۲. شمایی از معماری Xen [۵۱]

۳,۲,۱. بررسی اجزاء معماری Xen

در این بخش اجزاء تشکیل دهنده معماری Xen که به طور شماتیک در شکل ۳,۲ آمده است را بیشتر بررسی می کنیم.

۳,۲,۱,۱. Xen Hypervisor

این بخش اولین لایه مجرد نرم افزار مجازی سازی است که مستقیماً بر روی سخت افزار و زیر همه سیستم های عامل قرار گرفته است. Xen Hypervisor مسئول زمان بندی پردازنده و مدیریت حافظه برای ماشین های مجازی است که روی آن در حال اجرا هستند. همچنین این بخش هیچ گونه درکی از پروتکل های شبکه، دستگاه های ذخیره سازی خارجی، ویدئو و دیگر ورودی/خروجی های استاندارد ماشین ندارد.

۳,۲,۱,۲. Domain 0

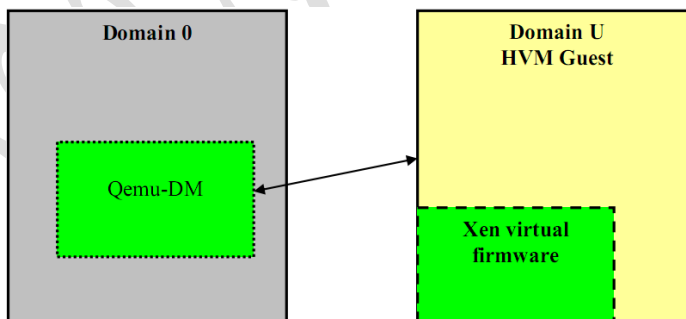
در مقابل پارتیشن های موجود در Hyper-V که هر یک مسئول نگهداری یک سیستم عامل بودند، در اینجا چندین دامنه وجود دارد. دامنه صفر که در واقع یک هسته تغییر یافته لینوکس است، تنها ماشین مجازی در حال اجرا بر روی Xen است که دسترسی مخصوصی برای کار با ورودی/خروجی های استاندارد و نیز ارتباط با دیگر ماشین های مجازی دارد. بر روی هر مجازی ساز Xen قبل از اجرای هر سیستم عامل دیگری لازم است دامنه صفر اجرا گردد. در دامنه صفر دو راه انداز برای پشتیبانی درخواست های سیستم های عامل میهمان وجود دارد. یکی راه انداز شبکه، برای ارتباط مستقیم با سخت افزار شبکه و دیگری راه انداز دیسک، برای کار با دیسک محلی و ذخیره و بازیابی اطلاعات بر حسب نیاز سیستم های عامل [۵۱].

Domain U ۳,۲,۱,۳

به هر ماشین مجازی در Xen یک دامنه^۱ U گفته می شود. همچنین تمام ماشین های مجازی که به صورت ضمنی مجازی سازی شده اند، به عنوان Domain U PV Guests و ماشین هایی که به صورت کامل مجازی سازی شده اند به عنوان Domain U HVM Guests شناخته می شوند.

لازم به ذکر است که راه اندازی ماشین های HVM بر روی Xen در صورتی امکان پذیر است که پردازنده از تکنولوژی مجازی سازی سخت افزاری^۲ پشتیبانی نماید. این امکان در پردازنده هایی مانند اینتل سری VT-x و پردازنده های AMD سری AMD-V وجود دارد [۵۲].

هر یک از ماشین های PV شامل دو راه انداز برای شبکه و دیسک هستند در حالی که ماشین های HVM این راه انداز های را ندارند و در عوض برای هر یک از آنها یک دیمون^۳ در دامنه صفر به نام Qemu-dm مسئول برقراری ارتباط سیستم عامل میهمان (دامنه U) با کارت شبکه و دیسک فیزیکی است. در حالت دوم، لازم است نرم افزاری به نام Xen virtual firmware بر روی سیستم عامل میهمان اجرا شود تا بتواند BIOS را برای راه اندازی سیستم عامل شبیه سازی نماید (شکل ۳,۳).



شکل ۳,۳. نحوه سرویس دهی به ماشین میزبان توسط Qemu-DM [۵۱]

¹ Domain Unprivileged

² hardware virtualization

³ Daemon

۳,۲,۱,۴ Xenstored

این دیمون مسئول نگهداری یک رجیستری از اطلاعات حافظه و وضعیت کانال های ارتباطی بین دامنه صفر و دامنه های U می باشد. ماشین مجازی مستقر در دامنه صفر با کنترل این رجیستری، ارتباط دیگر ماشین های مجازی را با منابع سخت افزاری مدیریت می کند.

۳,۲,۲ ارتباطات مابین دامنه صفر و دامنه های U

همانطور که قبلاً گفته شد، Xen Hypervisor قادر به ارائه سرویس شبکه و دیسک به سیستم های عامل میهمان نمی باشد بنابراین این سیستم های عامل باید از طریق hypervisor با سیستم عامل موجود در دامنه صفر ارتباط برقرار کنند تا سرویس مورد نظر خود را از وی بگیرد. برای مطالعه بیشتر در مورد Xen به [۵۳] الی [۵۸] مراجعه کنید.

۳,۳ VMware ESXi

این نرم افزار مجازی سازی که یکی از معروفترین hypervisor های موجود بوده و بیشترین سهم بازار مجازی سازی را نیز در اختیار دارد، امکان پیاده سازی مجازی سازی کامل را به کاربر داده و بنابراین سیستم های عامل میهمان کاملاً مستقل از hypervisor و بی اطلاع از آن کار می کنند. همچنین این hypervisor بر خلاف دو نمونه قبلی نیازی به اجرای سیستم عامل میزبان ندارد و مستقیماً بر روی سخت افزار نصب می گردد.

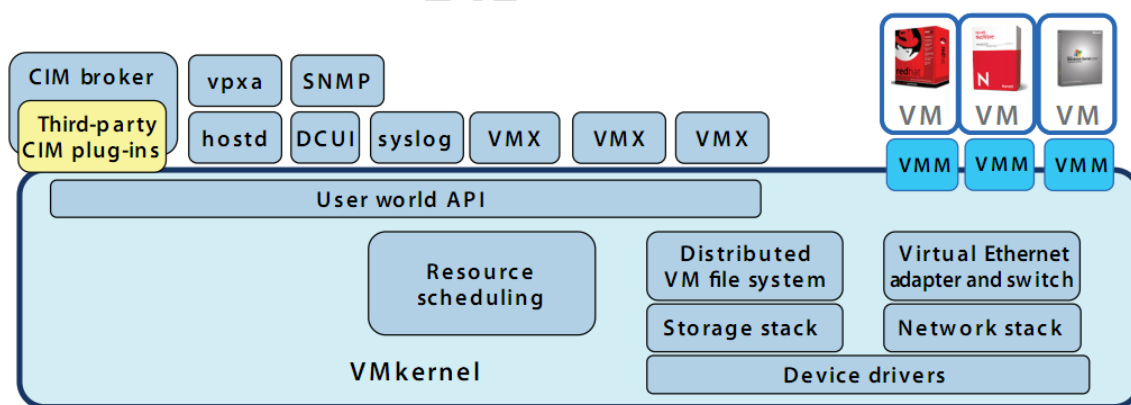
در ادامه به تشریح کلیات معماری آن خواهیم پرداخت [۳] و [۵۹].

این نرم افزار نیز مانند نمونه های قبلی از یک ساختار لایه ای پیروی می کند. در پایین ترین بخش، یک هسته تغییر یافته لینوکس به نام VMkernel قرار دارد که به طور مستقیم بر روی سخت افزار قرار گرفته است. وظیفه این هسته مدیریت همه پروسه ها از جمله برنامه های مدیریتی،

agentها و نیز ماشین های مجازی است. همچنین این بخش مدیریت سخت افزارها و منابع ماشین فیزیکی را بر عهده دارد. در واقع این هسته تمام وظایف یک سیستم عامل از جمله ایجاد پروسه ها، کنترل و سیگنال، مدیریت فایل سیستم و threadها را برای ماشین انجام می دهد. همچنین به طور اخص، VMkernel برای پشتیبانی از اجرای چندین ماشین مجازی طراحی شده است. از اصلی ترین پروسه هایی که بر روی VMkernel اجرا می شوند میتوان به موارد زیر اشاره کرد:

- رابط مستقیم کاربر (DCUI)^۱ که یک رابط سطح پایین برای مدیریت و پیکربندی hypervisor است و یک ارتباط کنسولی برای پیکربندی اولیه در اختیار می گذارد.
- VMM^۲ پروسه ای است که محیطی اجرایی برای ماشین مجازی ایجاد می کند. هر ماشین مجازی، VMM و پروسه کمکی VMX مخصوص به خود را دارد.
- Agent هایی که وظیفه ارائه سرویس های سطح بالا مانند HA را بر عهده دارند.
- CIM^۳ رابطی است که به کمک API های استاندارد، امکان مدیریت و نگهداری سخت افزار را به صورت راه دور فراهم می کند.

شکل ۳،۴ شمایی کلی از معماری گفته شده را نشان می دهد.



شکل ۳،۴. شمایی از معماری VMware ESXi [۳]

^۱ Direct Console User Interface

^۲ virtual machine monitor

^۳ Common Information Model

۳,۳,۱. بررسی اجزاء معماری VMware ESX

در این بخش به بررسی دقیقتر اجزاء سازنده VMware ESX خواهیم پرداخت.

۳,۳,۱,۱. VMkernel

این هسته در واقع یک سیستم عامل از خانواده POSIX است که توسط شرکت VMware طراحی شده و همه وظایف یک سیستم عامل مانند ایجاد و مدیریت پروسه ها، سیگنال ها و فایل سیستم و مدیریت I/O را انجام می دهد. همچنین این هسته به طور اختصاصی برای مدیریت چندین ماشین مجازی طراحی شده است و وظایف زیر را برای آنها انجام می دهد:

- زمانبندی منابع
- مدیریت پشته های I/O
- مدیریت راه انداز سخت افزارها

۳,۳,۱,۲. File System

VMkernel از فایل سیستم ساده ای برای نگهداری فایل های پیکربندی ESX، فایل های Log و بسته های اجرایی استفاده می کند. به عنوان مثال، همانند لینوکس، فایل های پیکربندی در مسیر /etc/vmware و فایل های log در مسیر /var/log/vmware ذخیره می شود. این فایل سیستم مستقل از فایل سیستم VMFS است که برای نگهداری ماشین های مجازی به کار می رود. منابع ذخیره سازی VMFS ممکن است بر روی دیسک محلی سرور یا یک دستگاه ذخیره سازی مشترک قرار داشته باشد. اگر منابع ذخیره سازی VMFS در خارج از سرور باشد، نیازی به وجود دیسک محلی نخواهد بود زیرا خود ESX نیز بدون نیاز به دیسک و به صورت embedded از روی یک حافظه flash قابل بارگزاری است.

۳,۳,۱,۳. CIM^۱

¹ Common Information Model

این مدل در واقع یک استاندارد باز است که چگونگی ارائه و مدیریت منابع پردازشی را در یک ESX تعریف می کند. به عبارت دیگر CIM چارچوبی را برای مانیتورینگ استاندارد و بدون نیاز به مفهوم agent بر روی سخت افزار ارائه می دهد. و شامل مدیر شیء CIM^۱ و ارائه کننده CIM^۲ است.

ارائه کننده های CIM مکانیزم های رابط برای مدیریت و دسترسی hypervisor به راه انداز دستگاه ها و دیگر سخت افزارهای سیستمی هستند. تولید کنندگان سخت افزار می توانند با تولید این ارائه کننده ها، امکان استفاده ESX از راه انداز سخت افزار خود را فراهم سازند و به این ترتیب ESX توانایی پشتیبانی و کار با هر نوع سخت افزار جدیدی را نیز دارد. همچنین ESX از این مکانیزم ها برای مانیتورینگ سخت افزار سرور، زیر ساخت ذخیره سازی ESX و منابع مخصوص مجازی سازی استفاده می کند. همچنین باید توجه داشت که این ماژول ها به دلیل توکار بودن در ESX، بسیار سبک و کاربردی هستند.

از سوی دیگر، مدیر شیء CIM، رابطی است که توسعه دهندگان می توانند ماژول های جدید ارائه کننده CIM را نوشته و به ESX اضافه کنند. البته باید توجه داشت که این بسته ها نمی توانند در حال اجرا به ESX اضافه شوند و باید با آن packaged شوند.

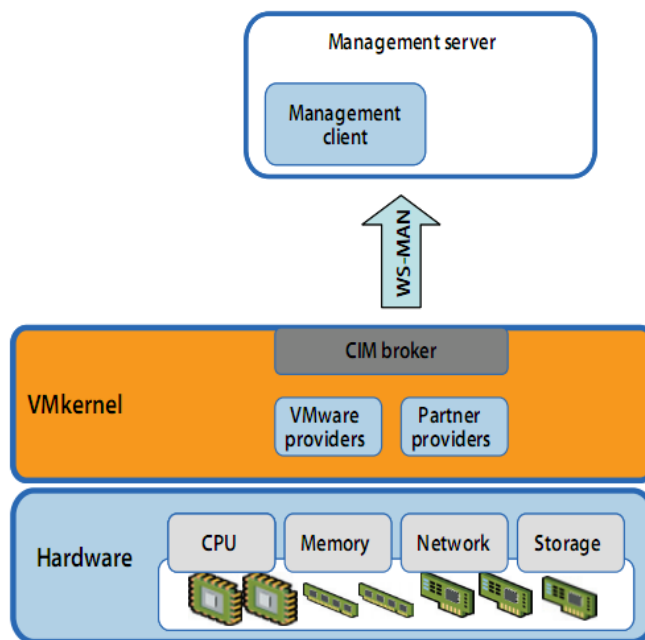
رابط CIM^۴ اطلاعات را از همه ارائه کننده های CIM جمع آوری کرده و از طریق API های استاندارد مانند WS-WAM به محیط مدیریتی منتقل می کند. شکل ۳،۵ ساختار مدیریت CIM را نشان می دهد.

^۱ CIM object manager (CIM broker)

^۲ CIM provider

^۳ Embedded

^۴ CIM broker



شکل ۳,۵. ساختار شماتیک مدیریت CIM [۳]

۳,۴. طراحی یک معماری برای دیتا سنتر

پس از بررسی نرم افزارهای hypervisor مطرح در بازار و مقایسه آنها، در این بخش با انتخاب یکی از آنها به طراحی یک معماری برای ساخت دیتا سنتر می پردازیم. این معماری پایه طراحی مدل فرمال در فصل بعد خواهد بود.

لازم به یادآوری است که hypervisor های معرفی شده در بخش قبل، با نصب بر روی یک ماشین فیزیکی، به مدیر سیستم اجازه اجرای چندین سیستم عامل را به طور همزمان می داد که هر یک از آنها در یک فضای مجزای حافظه به عنوان یک ماشین مجازی اجرا می شدند. در این مرحله با فرض اینکه به جای یک ماشین فیزیکی، چندین ماشین فیزیکی مشابه وجود دارد، لازم است طراحی برای مجازی سازی یکپارچه این ماشین ها ارائه گردد. به طوری که بر روی این ماشین های مجازی

اهدافی از جمله HA، انتقال ماشین ها و سرویس های در حال کار^۱ مدیریت یکپارچه ماشین های مجازی، مدیریت یکپارچه منابع ذخیره سازی مشترک، مدیریت ساختار شبکه و مدیریت یکپارچه منابع پردازشی حاصل گردد.

در مقایسه hypervisor های معرفی شده با یکدیگر، با توجه به مشخصه های هریک، در این تحقیق از معماری VMware ESX استفاده خواهد شد. برخی از دلایل این انتخاب عبارت است از:

۱- VMware ESX نیاز به نصب سیستم عامل میزبان و نیز پارتیشن ریشه یا دامنه صفر برای نگهداری یک رابط سیستم عاملی ندارد. بنابر این به نظر می رسد کاراتر و کم هزینه تر است.

۲- در معماری VMware ESX ماشین های مجازی برای استفاده از منابع سخت افزاری به طور مستقیم با VMkernel ارتباط برقرار می کنند و رابط سیستم عاملی در این بین وجود ندارد. بنابراین سیستم های عامل میهمان کاملاً مستقل و بدون اطلاع از محیط مجازی کار خواهند کرد. این امر خطاپذیری ماشین های مجازی را کم کرده و نیز به مدیر اجازه می دهد هر نوع سیستم عاملی را بر روی این پلتفرم نصب و راه اندازی نماید. در این میان فقط محدودیت های سخت افزاری مانند معماری ماشین فیزیکی مطرح می باشد.

۳- با توجه به منحصر بودن حوزه تخصصی شرکت VMware به تکنولوژی های مجازی سازی و تجاری بودن محصولات آن، می توان راه کارهای ساخت یافته و پخته ای^۲ را برای طراحی معماری های گسترده تر مجازی مانند ترکیب چندین دیتا سنتر، ساخت دیتا سنترهای مجازی و در نهایت طراحی معماری های پردازش ابری^۳ که مبتنی بر این محصولات باشند یافت. بنابر این راه برای توسعه معماری های مبتنی بر VMware همیشه باز خواهد بود.

^۱ Live migration

^۲ Mature

^۳ Cloud Computing

با توجه به توضیحات ارائه شده، در ادامه بحث به معرفی یک معماری عمومی برای دیتا سنترها بر اساس پلتفرم VMware می پردازیم [۶۰].

همچنین لازم به یاد آوری است که فرض این تحقیق بر وجود تعدادی ماشین های فیزیکی همگون، مانند سرورهای HP Blade در یک محل فیزیکی است که قرار است بر روی آن به کمک ساختار مناسب، تعداد دلخواهی از ماشین های مجازی قرار گیرد و سرویس های مورد نیاز در یک دیتا سنتر با اندازه کوچک ارائه گردد. معماری دیتا سنترهای مجازی و پردازش ابری خارج از موضوع این بحث می باشد.

۳.۵. مختصری درباره vSphere

vSphere به عنوان سیستم عامل پردازش ابری شناخته می شود و در واقع مجموعه ای از سرویس ها است که به کمک ESX معماری یک دیتا سنتر را پیاده سازی می نماید. لازم به توضیح است که مفاهیم میزبان، کلاستر و مخزن منابع، شیوه ای پویا و انعطاف پذیر برای سامان دهی منابع پردازشی و ارتباط آنها با زیر ساخت منابع فیزیکی ارائه می دهد. یک میزبان به مجموع منابع پردازشی و حافظه یک سرور فیزیکی با معماری x86 اشاره می کند. برای مثال اگر یک سرور فیزیکی دارای چهار پردازنده دو هسته ای 4GHz و 8GB حافظه باشد، این سیستم دارای 32GHz قدرت پردازش و 32GB حافظه می باشد که برای اجرای ماشین های مجازی آماده است.

یک کلاستر مانند یک موجودیت واحد کار می کند و مدیریت می شود. این موجودیت منابع پردازشی و حافظه یک گروه از سرور های فیزیکی x86 را که از منابع مشترک شبکه و دیسک استفاده می کنند یکپارچه می نماید. به عنوان مثال، اگر یک گروه هشت تایی از سرورها با چهار

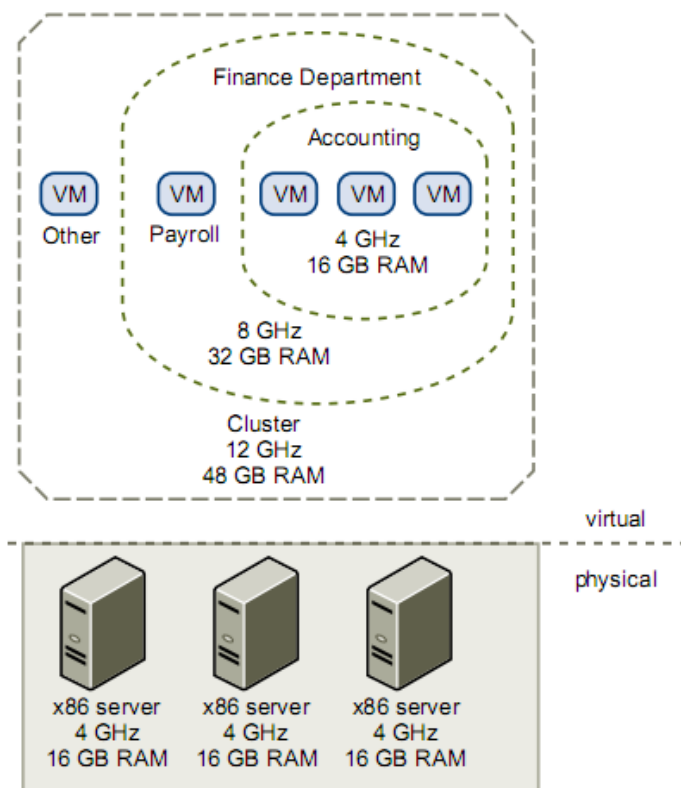
¹ host

² Resource pool

پردازنده دو هسته ای و 32B حافظه بر روی هر یک در اختیار باشد، کلاستر منبع یکپارچه ای شامل 256GHz قدرت پردازنده و 256GB حافظه برای اجرای ماشین های مجازی در اختیار کاربران قرار می دهد.

مخزن منابع بخشی از منابع پردازشی و حافظه یک میزبان و یا یک کلاستر است. این مخازن می توانند سلسله مراتبی و تودرتو باشند. مدیر سیستم می تواند یک مخزن را به مخازن کوچکتری تقسیم نماید.

در شکل ۳,۶ نمونه ای از مفاهیم گفته شده آمده است. سه سرور x86 که هر یک 4GHz توان پردازشی و 16GB حافظه دارند در نظر گرفته شده است. پس در مجموع یک کلاستر با توان پردازشی 12GHz و حافظه 48GB در اختیار داریم. مخزن پردازشی دپارتمان مالی، به میزان 8GHz از توان پردازشی و 32GB از حافظه این کلاستر را رزرو کرده است. همچنین 4GHz پردازنده و 16GB از حافظه برای ماشین های مجازی دیگر کنار گذاشته شده است. در کلاستر دپارتمان مالی، یک مخزن منابع کوچکتر شامل 4GHz از منابع پردازشی و 16GB حافظه برای ماشین های مجازی دپارتمان حسابداری ایجاد شده است. همچنین منابعی شامل 4GHz منبع پردازشی و 16GB حافظه به ماشین مجازی با عنوان payroll اختصاص یافته است.



شکل ۳,۶. مثالی از مفاهیم میزبان، کلاستر و مخزن منابع [۶۰]

مدیر سیستم می تواند به صورت پویا سیاست تخصیص منابع را عوض نماید. به عنوان مثال با زیاد شدن بار پردازشی یکی از ماشین های مجازی، می توان با قرض گرفتن منابع دیگر مخازن، منابع موجود در مخزن پردازشی مربوط به آن را افزایش داد و برای این کار نیازی به خاموش شدن هیچ یک از ماشین های مجازی وجود ندارد.

در ادامه به شرح سرویس هایی که معماری vSphere را می سازند می پردازیم [۶۰] و [۶۱].

۳,۵,۱. سرویس های زیر ساختی vSphere

- VMware vCompute
- VMware vStorage
- VMware vNetwork

۳.۵.۲. سرویس های کاربردی vSphere

سرویس هایی که خصوصياتی از سیستم مانند دسترسی پذیری، امنیت، توسعه پذیری و غیره را در سیستم فراهم می کنند. نمونه هایی از آن HA و تحمل خطا در سیستم می باشد.

۳.۵.۲.۱. سرور VMware vCenter

سروری که یک نقطه دسترسی واحد برای مدیریت و کنترل کلیه منابع و امکانات دیتا سنتر فراهم می کند. سرویس هایی مانند کنترل دسترسی، مشاهده وضعیت کارکرد سیستم و پیکر بندی سیستم نمونه هایی از سرویس های این سرور هستند.

۳.۵.۲.۲. Client ها

کاربران می توانند به کمک رابط های کاربری به منابع vSphere دسترسی داشته باشند. این ابزار شامل vSphere Client و رابط های تحت وب هستند.

۳.۵.۲.۳. ESX

همانطور که قبلا توضیح داده شد، یک نرم افزار مجازی ساز از کلاس hypervisor است که منابع سیستم را برای استفاده ماشین های مجازی مدیریت می کند.

۳.۵.۲.۴. vCenter Server

سروری برای مدیریت جامع منابع دیتا سنتر و پیکر بندی و مانیتورینگ سیستم می باشد.

۳.۵.۲.۵. VMFS^۱

یک فایل سیستم کارآمد برای مدیریت منابع ذخیره سازی کلاستر می باشد.

^۱ Virtual Machine File System

SMP^۱,۳,۵,۲,۶

ماژولی که به یک ماشین مجازی امکان استفاده از چندین پردازنده فیزیکی را به طور همزمان می دهد.

VCMS^۲,۳,۵,۲,۷

یک نقطه دسترسی مرکزی برای پیکربندی، تخصیص و مدیریت منابع زیر ساخت مجازی است.

VI Client^۳,۵,۲,۸

رابطی که امکان دسترسی از راه دور به VCMS و یا هر یک از ESX های نصب شده بر روی ماشین ها را برای کاربران و مدیر سیستم فراهم می نماید.

VMware VMotion^۳,۵,۲,۹

ماژولی که امکان انتقال بلادرنگ^۴ ماشین های مجازی از یک ماشین فیزیکی به ماشین فیزیکی دیگر را فراهم می کند. این انتقال بدون قطع سرویس و از کار افتادگی^۵ خواهد بود. برای انتقال یک ماشین مجازی، VMotion یک کپی از ماشین مجازی مذکور را بر روی یک سرور دیگر روشن می کند و وضعیت جاری پروسه ها را به آن انتقال می دهد. این عمل باعث انتقال بلادرنگ ماشین و عدم قطعی در سرویس های آن خواهد شد.

Storage VMotion^۳,۵,۲,۱۰

این سرویس امکان انتقال فایل های یک ماشین مجازی را از روی یک سرور فیزیکی به سرور فیزیکی دیگر فراهم می کند. این سرویس می تواند دیسک های مجازی و فایل های پیکربندی یک

¹ Virtual Symmetric Multi-Processing

² Virtual Center Management Server

³ Virtual Infrastructure Client

⁴ Live Migration

⁵ Downtime

ماشین در حال کار را به یک datastore جدید منتقل کند. همچنین این عمل را بدون قطعی و تاخیر در ماشین مجازی انجام می دهد.

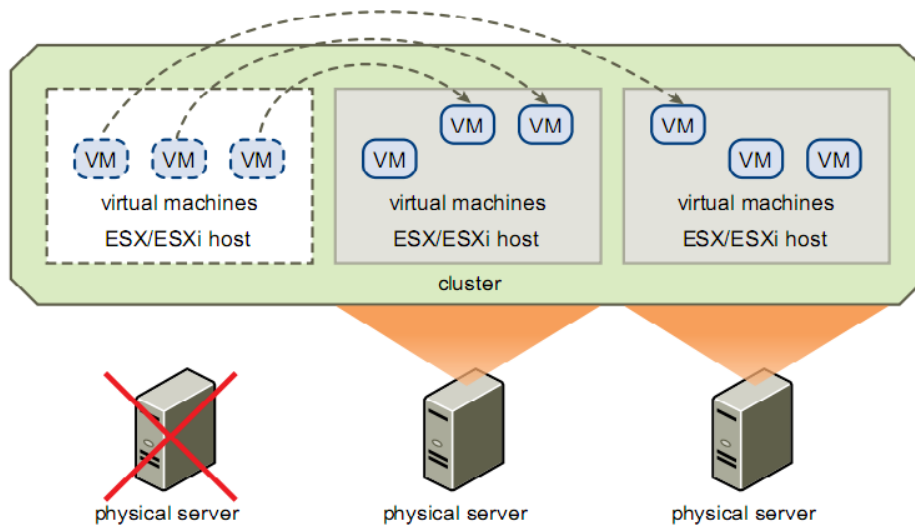
به کمک دو سرویس بالا مدیر سیستم می تواند ماشین مجازی و دیسک های آن را در یک محل قرار دهد و یا در سرور های مختلف توزیع نماید.

۳،۵،۲،۱۱. VMware HA^۱

ماژولی که امکان HA را با هزینه کم و به شکل ساده برای نرم افزارهای در حال اجرا بر روی ماشین های مجازی فراهم می کند. در صورت بروز یک اختلال، ماشین مجازی متاثر، به صورت خودکار بر روی سرور دیگری که دارای منابع اضافی است راه اندازی می گردد.

این سرویس تمام میزبانان روی هر کلاستر HA را مانیتور کرده و خرابی ها را شناسایی می کند. یک agent بر روی هر میزبان فیزیکی مسئول تولید یک سیگنال ضربانی است. این ضربان با ضربان دیگر میزبانان حاضر بر روی مخزن منابع همگام است. در صورت قطع شدن این ضربان بر روی هر یک از میزبانان، سرویس HA پروسه راه اندازی ماشین های مجازی موجود بر روی میزبان فیزیکی قطع شده را بر روی میزبانان دیگر اجرا می کند. البته این سرویس همواره اطمینان حاصل می کند که منابع پردازشی کافی بر روی مخزن منابع برای راه اندازی مجدد ماشین های مجازی در صورت بروز خطا وجود دارد. شکل ۳،۷ طرز کار سرویس HA را نشان می دهد.

^۱ VMware High Availability



شکل ۳،۷. طرز کار سرویس HA [۶۱]

با توجه به وظیفه ماژول HA، این سرویس ابزاری برای مانیتورینگ وضعیت ماشین های مجازی در کلاستر نیز ارائه می نماید. این امکان می تواند در صورت بروز خرابی بر روی یک میزبان، مراتب را گزارش نموده و نسبت به پروسه راه اندازی ماشین های مجازی متاثر شده اقدام کند.

۳،۵،۲،۱۲. DRS^۱

این ماژول به صورت هوشمند و پویا منابع سخت افزاری سیستم را به ماشین های مجازی تخصیص داده و بالانس می کند.

در واقع این سرویس یک کلاستر از سرورهای فیزیکی را به صورت منابع یک کامپیوتر واحد مدیریت می کند. وقتی مدیر یک ماشین مجازی را به یک کلاستر نسبت می دهد، DRS یک سرور فیزیکی مناسب را برای اجرای آن پیدا می کند. همچنین DRS ماشین های مجازی را به شکلی بر روی سرور های فیزیکی توزیع می کند که بالانس بودن بار پردازشی و اعمال سیاست گزاری تخصیص منابع از جمله رزور منابع، اولویت بندی و محدودیت ها اطمینان حاصل نماید. وقتی یک ماشین مجازی روشن می شود، DRS یک بررسی اولیه برای جاگزاری آن بر روی یک سرور فیزیکی انجام

^۱ Distributed Resource Scheduler

می دهد. اگر شرایط یک کلاستر مانند بار پردازشی و منابع در دسترس تغییر کند، DRS به کمک سرویس های VMotion ماشین های مجازی را به سرور فیزیکی جدیدی منتقل می کند. همچنین اگر یک سرور فیزیکی به مجموعه اضافه شود، DRS به ماشین های مجازی امکان استفاده از منابع پردازشی و حافظه سرور جدید را می دهد.

DRS شامل سرویس¹ DPM است که امکان صرفه جویی محسوسی را در مصرف انرژی توسط دیتا سنتر فراهم می کند. در صورت فعال بودن DPM، سیستم، ظرفیت منابع را در سطح کلاستر و میزان، با میزان منابع مورد نیاز ماشین های مجازی مقایسه می کند. اگر این نیاز ماشین ها با تعداد کمتری از سرور های فیزیکی قابل سرویس دهی باشد، DPM ماشین های مجازی را به این تعداد سرور منتقل و بقیه سرورها را خاموش می کند. وقتی تقاضا برای منابع دوباره افزایش یافت DPM سرور های مذکور را دوباره روشن کرده و ماشین های مجازی را به محل اولیه بر می گرداند. با این مکانیزم تخصیص منابع به کلاسترها فقط به اندازه نیاز آنها، میزان مصرف انرژی توسط سرور های فیزیکی به شکل قابل ملاحظه ای کاهش می یابد.

۳,۵,۲,۱۳ Consolidated Backup

ابزاری راحت و متمرکز برای گرفتن backup از ماشین های مجازی است که بدون نیاز به agentها پیاده سازی شده است. این ماژول مدیریت backup را ساده تر کرده و بار پردازشی بر روی ESX را کم می کند.

۳,۵,۲,۱۴ vSphere SDK

رابطی استاندارد برای نوشتن ماژول های جانبی نرم افزاری و بر روی vSphere می باشد.

۳,۵,۲,۱۵ Fault Tolerance

¹ distributed power management

اگر این امکان برای یک ماشین مجازی فعال باشد، یک کپی از ماشین اولیه تحت عنوان ماشین ثانویه ایجاد می گردد. همچنین تمام اعمالی که بر روی ماشین مجازی اصلی انجام می گیرد بر روی کپی آن نیز اعمال می شود. در صورت بروز خطا و غیر قابل استفاده شدن ماشین مجازی، کپی آن فعال خواهد شد و کار ماشین را ادامه می دهد.

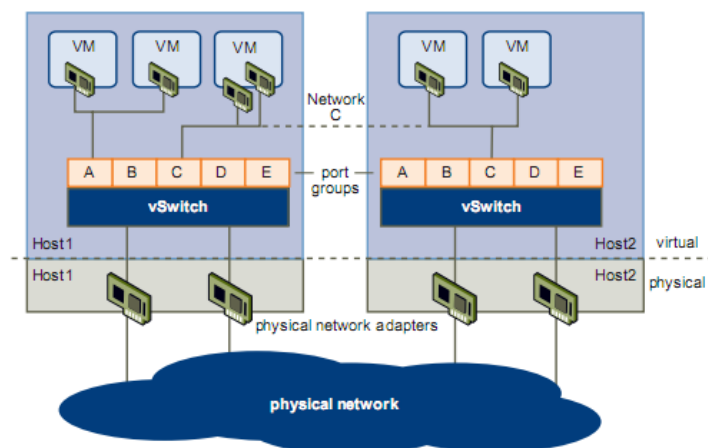
۳,۵,۳. سرویس های توزیع شده در vSphere

از بین سرویس های توصیف شده در بالا، تعدادی از آنها به صورت توزیع شده بر روی معماری انجام وظیفه می کنند. این سرویس ها امکان مدیریت و تخصیص منابع را به صورت کار آمد و خودکار فراهم می کنند و شامل سرویس های زیر هستند:

- VMware VMotion
- VMware Storage VMotion
- VMware DRS
- VMware HA
- Fault Tolerance

۳,۵,۴. معماری شبکه

Hypervisor با شبیه سازی مکانیزم های فیزیکی دسترسی به شبکه در محیط مجازی، همه امکانات این ابزار و نیز امکانات دیگری که در محیط واقعی قابل دسترسی نیست را برای ماشین های مجازی فراهم می کند.



شکل ۳,۸. شمایی از معماری شبکه در محیط مجازی [۶۰]

همانطور که در شکل ۳,۸ دیده می شود، hypervisor با استفاده از کارت شبکه های مجازی، سوئیچ های مجازی و port group ها امکان ایجاد vNIC به تعداد دلخواه، برقراری ارتباط vNIC با کارت شبکه فیزیکی، پشتیبانی از پروتکل های سوئیچینگ لایه ۲ و قرار دادن تعداد دلخواهی از vNIC ها در یک شبکه مجزا را فراهم می کند.

در این معماری، vSwitch شبیه یک سوئیچ فیزیکی لایه ۲ عمل می کند و هر ماشین فیزیکی vSwitch مربوط به خود را دارد. این سوئیچ از یک طرف با port group متصل به ماشین های مجازی مرتبط بوده و از طرف دیگر یک uplink با کارت شبکه های فیزیکی ماشین دارد. همچنین این سوئیچ می تواند به بیش از یک کارت شبکه فیزیکی متصل شود امکاناتی از جمله NIC teaming را فراهم کند [۶۲].

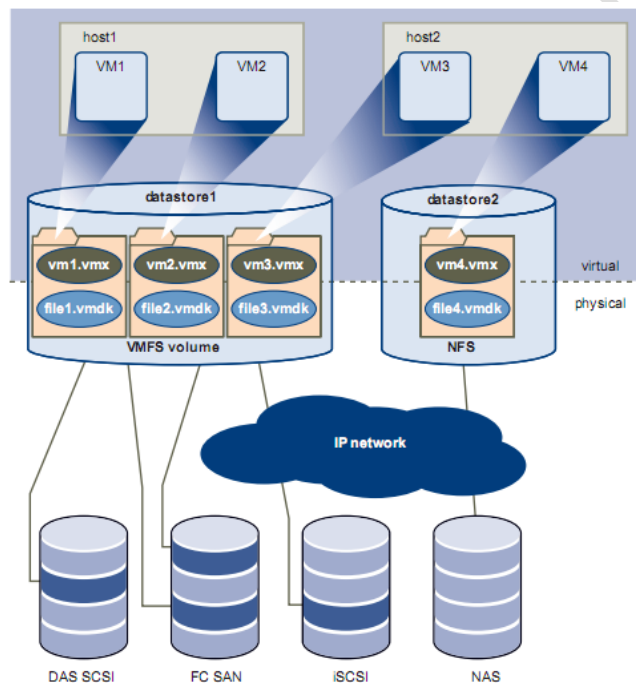
همچنین port group مفهومی منحصر به محیط مجازی است که به کمک آن می توان مفهوم VLANing را بر روی vNIC ها پیاده سازی نمود. به این معنی که با قرار دادن چندین vNIC در یک port group، می توان آنها را به صورت یک شبکه مجزا برای محیط مجازی معرفی کرد و قوانین و سیاست های مشترکی را بر روی آنها اعمال نمود.

¹ vNIC

² vSwitch

۳,۵,۵. معماری محل ذخیره سازی داده ها

این معماری شامل لایه هایی برای ایجاد انتزاع کافی و پوشاندن و مدیریت پیچیدگی ابزار ذخیره سازی متنوع است که می تواند یک عنصر ذخیره سازی ساده و یکسان را در اختیار ماشین های مجازی قرار دهد. بنابراین همه نرم افزارها و سیستم های عامل در حال اجرا، یک دیسک ساده SCSI که به Bus logic مجازی متصل است را مشاهده و با آن کار می کنند [۶۱].



شکل ۳,۹. شمایی از معماری ذخیره سازی [۶۰]

دیسک های مجازی توسط ماژول Datastore به ماشین های مجازی ارائه می شوند. در واقع، datastore یک نرم افزار ذخیره سازی است که که سرویس ذخیره سازی را برای ماشین های مجازی فراهم می کند همانطور که در شکل ۳,۹ دیده می شود، خود ماشین مجازی نیز به عنوان یک فایل در پوشه مربوط به خود ذخیره می شود (vm1.vmx) که این فایل بعداً به صورت یک فایل بر روی دیسک فیزیکی نگهداری می گردد (vm1.vmdx). بنابراین سیستم عامل و دیسک

متعلق به آن به راحتی به عنوان یک فایل قابل جابه جایی، کپی برداری، backup و غیره است. در اینجا نیز دیسک های مجازی مانند کارت های شبکه مجازی قابل افزودن به یک ماشین مجازی در حال کار هستند^[۶۳].

بنابراین در این مدل، نرم افزارها بدون اطلاع از پیچیدگی تکنولوژی ذخیره سازی می توانند از آن استفاده کنند و در عین حال معماری می تواند از انواع تکنولوژی ها مانند Fiber Channel SAN، iSCSI SAN، Attached Storage Direct و NAS استفاده نمایند.

از نظر فیزیکی، هر datastore، یک پارتیشن از فایل سیستم VMFS و یا یک پوشه است که بر روی دیسک ذخیره شده است. هر datastore می تواند شامل چندین منبع ذخیره سازی فیزیکی باشد. هر پارتیشن VMFS شامل یک یا چند LUN^۲ است که مستقیماً بر روی دیسک های فیزیکی مانند iSCSI یا ... قرار گرفته اند. LUN های اضافه شده، به طور اتوماتیک شناسایی و به منابع سیستم اضافه می شوند. برای اضافه کردن این LUN ها به datastore های موجود، نیازی به خاموش کردن هیچ یک از بخش های سیستم نیست. همچنین اگر یکی از LUN ها دچار خرابی شود فقط ماشین های مجازی که به آن وابسته هستند از مدار خارج می شوند و بقیه ماشین ها به صورت عادی به کار خود ادامه می دهند.

VMFS یک فایل سیستم کلاستر برای مدیریت دیسک های مشترک و فراهم آوردن امکان دسترسی چندین سرور فیزیکی به این منابع است. همچنین VMFS با مکانیزم فکل توزیع شده دیسک برای هر ماشین مجازی، تضمین می کند که هر ماشین در یک زمان تنها بر روی یک سرور فیزیکی قابل اجرا است. در صورت خراب شدن یک سرور فیزیکی این فکل آزاد شده و ماشین های مجازی روی آن می توانند بر روی سرور دیگری شروع به کار نمایند.

VMFS همچنین مکانیزم هایی را در سطح کلان برای پیشگیری و برخورد با خرابی منابع ذخیره سازی و بازیابی اطلاعات ارائه می نماید که از آن جمله می توان به ژورنالینگ توزیع شده،

¹ Hot-adding

² Logical Unit Number

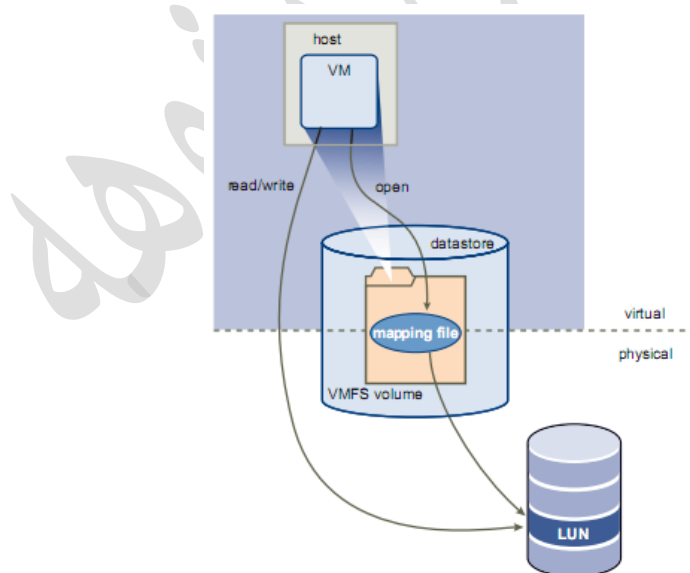
³ On-disk distributed locking

⁴ Enterprise

مکانیزم IO path برای جلوگیری از خرابی و ذخیره تصاویر لحظه ای از سیستم^۱ اشاره کرد. سیستم به کمک این ابزار می تواند مدت از کار افتادگی سرور های فیزیکی و منابع ذخیره سازی خود را به حداقل برساند.

از دیگر سرویس های VMFS می توان به نگاشت مستقیم دیسک یا RDM^۲ اشاره نمود. RDM مکانیزمی برای دسترسی مستقیم ماشین های مجازی به LUN های مستقر بر روی دیسک فراهم می کند. این راهکار در مواقعی از جمله موارد زیر بسیار مفید است:

- دریافت تصاویر لحظه ای از SAN و دیگر نرم افزارهای چند لایه که بر روی ماشین های مجازی نصب شده اند. که در این حالت RDM به کمک خصوصیات ذاتی SAN کارایی ذخیره سازی را بالا می برد.
- استفاده از سرویس کلاستر سازی میکروسافت^۳ که بر روی چندین سرور فیزیکی قرار گرفته و امکان کلاستر سازی ماشین های مجازی با مجازی و نیز ماشین های فیزیکی با مجازی را بر عهده دارد.



¹ Machine state snap-shots

² Raw Device Mapping

³ Microsoft Clustering Services

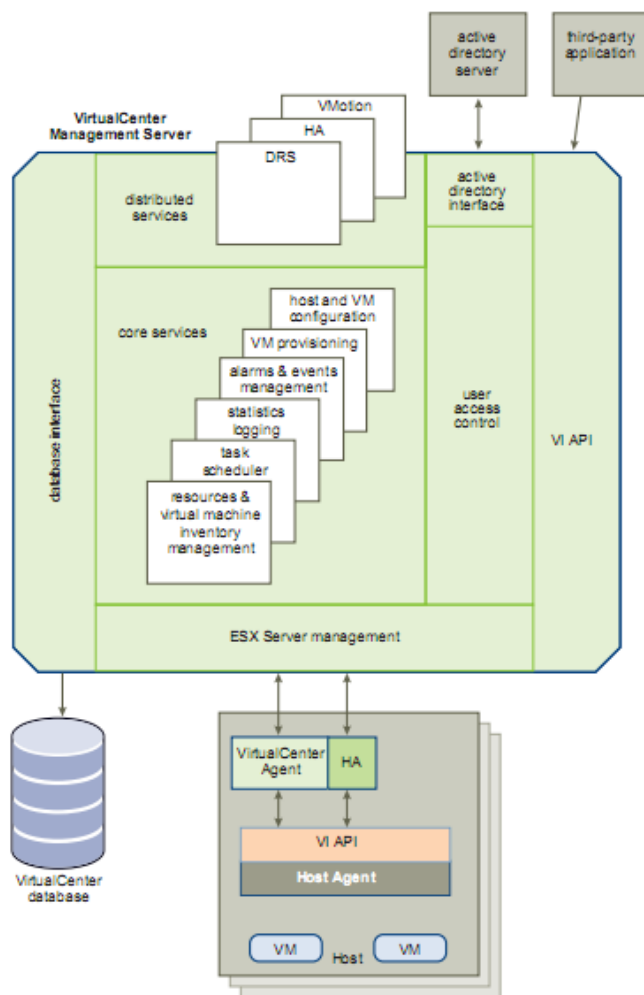
شکل ۳,۱۰. طرز کار RDM [۶۰]

مطابق شکل ۳,۱۰ یک RDM را می توان به عنوان یک اشاره گر^۱ در نظر گرفت که به یک LUN اشاره می کند. این نگاشت LUN را به صورت یک فایل در پارتیشن VMFS نمایش می دهد. وقتی LUN برای دسترسی خواندن یا نوشتن باز می شود، VMFS فایل RDM را برای تشخیص محل فیزیکی ذخیره سازی مورد نظر بررسی می کند. پس از فقل کردن LUN مورد نظر، دسترسی مناسب را برای انجام عملیات در اختیار نرم افزار قرار می دهد.

۳,۵,۶. معماری سرور مدیریت VirtualCenter

این سرور ابزاری برای مدیریت متمرکز منابع فیزیکی که ممکن است بر روی چندین ESX پراکنده باشد ایجاد می کند و به صورت پنبلی انعطاف پذیر و یکپارچه برای مدیریت منابع فیزیکی و مجازی در اختیار مدیر سیستم قرار می دهد [۶۴].

¹ Symbolic link



شکل ۳،۱۱. شمایی از ساختار سرور مدیریت VirtualCenter [۶۰]

در شکل ۳،۱۱ اجزای اصلی سرور نشان داده شده است.

۳،۵،۶،۱. User Access Control

این ماژول امکان تعریف کاربرانی با سطوح دسترسی مختلف را برای سیستم فراهم می کند. به عنوان مثال کاربری که توانایی مدیریتی سرور های فیزیکی را داشته باشد و یا کاربری که توانایی مدیریت تنها تعداد مشخصی از ماشین های مجازی را داشته باشد.

Core Service. ۳, ۵, ۶, ۲

سرویس پایه برای مدیریت دیتا سنتر مجازی است که سرویس هایی مانند Logging and Statistics, VM provisioning و Host and VM Configuration را ارائه می دهد. برای مطالعه بیشتر در مورد VMware به [۶۵] الی [۶۹] و [۷۲] مراجعه کنید.

۳, ۶. جمع بندی

در این فصل به تعریف نرم افزارهای Hypervisor و معرفی سه نمونه پیشرو در بازار پرداخته شد. این سه Hypervisor که در حال حاضر بخش بزرگی از بازار را در اختیار دارند عبارتند از VMware ESX و Xen، Microsoft Hyper-V.

در مرحله بعد به تشریح معماری هر یک پرداختیم تا نقاط ضعف و قوت شان را با یکدیگر مقایسه کنیم. در پایان و با توجه به اینکه در این تحقیق از محصول شرکت VMware برای طراحی لایه نرم افزاری دیتا سنتر استفاده خواهد شد، بر روی این تکنولوژی تمرکز بیشتری داشتیم. به این ترتیب که علاوه بر معماری ESX، به تشریح سرویس های سطح بالا که معماری VMware vSphere و یا سیستم های پردازش ابری را بر مبنای تکنولوژی VMware می سازند پرداختیم. این سرویس ها عبارت اند از VMotion، HA، Fault Tolerance، DRS و دیگر سرویس هایی که در ترکیب با بقیه کار می کنند.

در فصل بعد پس از طراحی یک نمونه دیتا سنتر کوچک، از بخش های مختلف آن مدل های فرمال تهیه می کنیم. برای تهیه مدل فرمال از زبان شبکه های پتری، تشریح شده در فصل ۲، کمک خواهیم گرفت.

فصل چهارم: ارائه مدل فرمال برای دیتا سنتر و تحلیل آن

در این فصل ابتدا یک دیتا سنتر نمونه تشریح می گردد و سپس این دیتا سنتر در لایه های مختلف توسط شبکه های پتری مدل خواهد شد. برای بررسی دقیقتر، سعی شده است که نمونه معرفی شده برای دیتا سنتر تا حد امکان ساده و در عین حال شامل تمام بخش های لازم برای یک دیتا سنتر واقعی باشد.

همچنین لازم به ذکر است که این تحلیل ساختار و مدل سازی تا سطح سرویس انجام شده است. به این معنی که از بررسی مکانیزم های داخلی ماشین های مجازی از جمله سیستم عامل میهمان و سرویس های آن و نیز ساختار منابع سخت افزاری از جمله پردازنده، حافظه و غیره چشم پوشی شده است. بنابراین به تشریح ساختار کلی لایه مجازی سازی، سرویس های سطح بالا از جمله HA، Fault Tolerance و vMotion و نیز شیوه کار معماری های شبکه و سیستم ذخیره سازی پرداخته شده است.

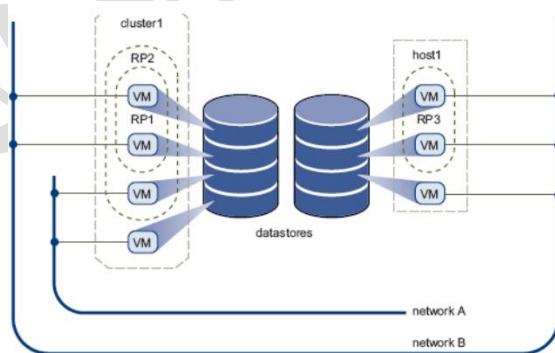
به عبارت دیگر از دیدگاه سطح دانه پذیری سیستم^۱ می توان گفت مدل ها با حداکثر انتزاع ممکن و در سطح اجزاء اصلی دیتا سنتر انجام شده است.

¹ Granularity

۴,۱. تشریح دیتا سنتر نمونه

در این فصل با تکیه بر ابزار و تکنولوژی های معرفی شده، تلاش می شود یک الگوی نمونه برای دیتا سنترهای نوین معرفی گردد. در این معماری، منابع پردازشی هر ماشین فیزیکی بخشی از مخزن منابع کل سیستم هستند. بنابراین مدیر سیستم می تواند این منابع را به هر شکل مورد نیاز تخصیص دهد. همچنین وی می تواند به سرعت بین ماشین ها سوئیچ کند، از آنها نسخه پشتیبان تهیه نماید، آنها را کپی کند و به صورت متمرکز آنها را مدیریت نماید.

زیر ساخت VMWare همه منابع پردازشی از جمله سرورها، منابع ذخیره سازی و شبکه را مجازی سازی می کند. این زیر ساخت با گردآوری همه منابع و نمایش مجموعه ای ساده شده و یکپارچه از آنها، مدیر را در درک بهتر ساختار موجود دیتا سنتر و مدیریت و تغییر آن یاری می رساند. به کمک این ساختار می توان منابع توزیع شده در یک دیتا سنتر را به صورت مجموعه ای از ابزار مشترک مدیریت نمود. همچنین می توان از دیتا سنتر در مصارف گوناگونی استفاده کرد بدون اینکه نگران گوناگونی سخت افزار و نوع مصرف آن باشیم. شکل ۴,۱ نحوه پیکربندی و معماری یک طرح نمونه VMware را نشان می دهد.



شکل ۴,۱. زیر ساخت دیتا سنتر مجازی [۶۳]

پیش از آنکه ساختار دیتا سنترها را دقیقتر بررسی نمائیم، لازم است برخی از اصطلاحات و مفاهیم را تشریح کنیم. این مفاهیم در بخش های مختلف لایه نرم افزاری در سیستم مذکور به کار می روند [۶۰] و [۶۳].

- میزبان

یک میزبان نمود مجازی سازی شده منابع پردازشی و حافظه یک ماشین فیزیکی است که در حال اجرا کردن VMware ESX می باشد. در واقع بدنه اصلی فیزیکی دیتا سنتر از مجموعه ای از ماشین های میزبان تشکیل شده که در حال اجرا کردن زیر ساخت سیستم عاملی Hypervisor هستند.

- کلاستر

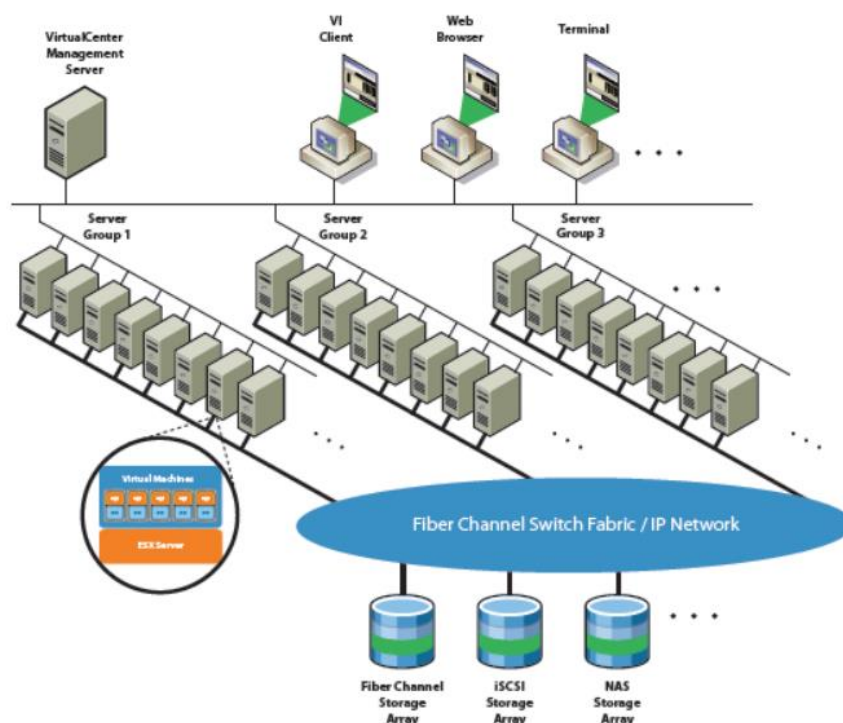
وقتی چند ماشین فیزیکی با یکدیگر همگروه می شوند و به صورت یک واحد یکپارچه مدیریت می گردند، منابع پردازشی و حافظه آنها که با یکدیگر جمع شده است یک کلاستر تشکیل می دهند. چنین گروهی از منابع پردازشی به منظور راه اندازی سرویس های سطح بالا مانند HA، Fault Tolerance و غیره مورد استفاده قرار می گیرند. می توان ماشین های فیزیکی جدیدی را به یک کلاستر اضافه یا از آن حذف نمود.

- مخزن منابع

مجموعه منابع پردازشی و حافظه ماشین هایی که در یک کلاستر قرار دارند، تشکیل مخزنی از منابع را می دهند که به همین نام شهرت یافته است. مدیر سیستم می تواند این منابع را به صورت سلسله مراتبی و یا گروه بندی شده به ماشین های مجازی تخصیص دهد و همچنین بخشی از آن را به صورت دستی و یا خودکار برای سرویس های سطح بالا که در یک دیتا سنتر ضروری است رزرو نماید.

در یک دیتا سنتر نمونه مانند شکل ۴،۲، چندین ماشین فیزیکی با معماری x86 در یک شبکه محلی به یکدیگر متصل شده اند. این شبکه ترجیها از قویترین و سریعترین انواع شبکه های در دسترس است زیرا در انتهای طراحی، نقاط اتصال سرورهای فیزیکی، گلوگاه ارتباطات دیتا سنتر خواهد بود. همچنین تعدادی منبع ذخیره سازی نیز در این شبکه وجود دارند. بر حسب تکنولوژی

موجود در بازار، این ابزار ذخیره سازی می تواند از دیسک های SCSI درون هر یک از سیستم های میزبان تا دیسک های SCSI مبتنی بر شبکه اترنت^۱ متنوع باشند. وجود تجهیزات تخصصی دیگر مانند سوئیچ های SAN برای برقراری ارتباط میان سرورهای فیزیکی و ابزار ذخیره سازی و نیز تجهیزات پشتیبان گیری^۲ نیز در دیتا سنتر عادی است.



شکل ۴،۲. یک الگوی نمونه برای زیر ساخت دیتا فیزیکی سنتر [۶۰]

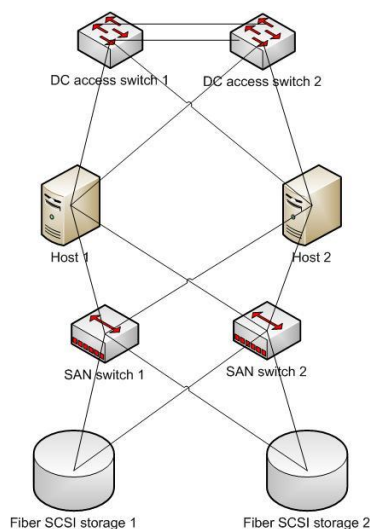
در این تحقیق برای طراحی و تحلیل مدل فرمال از یک دیتا سنتر، نمونه کوچکی از یک دیتا سنتر واقعی در نظر گرفته شده است که شامل دو سیستم میزبان و دو منبع ذخیره سازی می باشد. سیستم های میزبان به یک شبکه داخلی متصل هستند و به وسیله سوئیچ های SAN به دیسک های Fiber SCSI مرتبط شده اند. افزونگی^۳ در سیستم به منظور به حداقل رساندن قطعی و خرابی

^۱ Ethernet-based SCSI disks (iSCSI)

^۲ Backup systems

^۳ System redundancy

در شبکه زیر ساخت دیتا سنتر رعایت شده است. شمای کلی این ساختار را می توان در شکل ۴,۳ دید.



شکل ۴,۳. ساختار شماتیک دیتا سنتر نمونه

بعد از پیاده سازی سخت افزاری اجزا و نصب ESX بر روی سیستم های میزبان و نیز پس از پیکر بندی سیستم های میزبان و ذخیره سازی داده، سیستم می تواند شروع به کار نماید. در این مرحله انتظار می رود که همه چهار بخش اصلی سیستم یعنی دو سیستم میزبان و دو سیستم ذخیره سازی اطلاعات بدون مشکل بوت شده و آماده بارگذاری شوند. با این حال بروز اختلال در یکی یا بیشتر و آماده به کار نبودن بعضی از آنها در پایان مرحله بوت دور از انتظار نیست. در گام بعد، با توجه به بخش های آماده به کار، ماشین های مجازی و سرویس های سطح بالا می توانند اجرا و پیکر بندی شوند. با توجه به ساختار سرویس های سطح بالا و نحوه کارکرد آنها، واضح است که برای اجرای درست سرویس هایی مانند HA و FT حضور حداقل دو میزبان فیزیکی ضروری است [۶۱]. پس از بارگذاری ماشین های مجازی و سرویس های لازم، ESX شروع به دریافت درخواست های ماشین های مجازی و ارائه سرویس به آنها می کند. در کنار این روال دائمی، در صورت بروز مشکل در یکی از ماشین های میزبان یا منابع ذخیره سازی و یا درخواست مدیر برای انتقال یک

ماشین مجازی و غیره، سرویس متناظر می تواند فعال شده و وظیفه خود را انجام دهد. به عبارت دیگر می توان گفت که از این مرحله به بعد سیستم در یک وضعیت تعادل اجرایی مابین نحوه کار ESX و سرویس ها قرار می گیرد. در چنین تعادلی، مدیر می تواند ماشین های مجازی جدیدی ایجاد کند، ماشین های مجازی را از یک میزبان به میزبان دیگر منتقل نماید و نحوه تخصیص منابع به ماشین های مجازی و کلاسترها را تغییر دهد.

۴,۲. ارائه مدلی از رفتار کلی دیتا سنتر

در این بخش با مبنا قرار دادن طرح نمونه دیتا سنتر که در انتهای بخش قبل معرفی شد، سعی خواهیم کرد تا مدل سلسله مراتبی فرمالی از ساختار مذکور به دست آوریم. صفت "سلسله مراتبی"^۱ در تعریف فوق به این معنی است که در ابتدا با طرح یک مدل کاملاً کلی و انتزاعی، به تشریح رفتار کلی سیستم می پردازیم و سپس برای هر یک از بخش های پیچیده در رفتار سیستم یک مدل جداگانه طراحی می نمائیم که مدل های لایه دو، سه و ... محسوب می شوند. برای رسیدن به این منظور روی مفاهیم توضیح داده شده در فصل های گذشته تاکید خواهد شد که برای حفظ اختصار تکرار نشده اند.

با توجه به رفتار دیتا سنتر نمونه از زمان روشن شدن تا اجرایی شدن کامل آن، می توان مدلی کاملاً کلی و انتزاعی بدون بررسی جزئیات اجرایی مربوط به سرویس ها و مکانیسم های آن ارائه داد. نمونه ای از مدل طراحی شده در شکل ۴,۴ آمده است.

¹ Hierarchical

این مدل سلسله مراتبی می توان با طراحی یک مدل پتری برای هر گذار زمان دار، جزئیات بیشتری از سیستم را مدل کرده و بررسی نمود.

در ادامه به شرح گذارها و نحوه گردش کنترل در سیستم می پردازیم.

T0: سیستم به شکل فیزیکی روشن می شود.

T1: دستگاه ذخیره سازی ۱ با موفقیت بوت می شود

T2: دستگاه ذخیره سازی ۱ در مرحله بوت دچار مشکل شده و بوت نمی شود.

T3: دستگاه ذخیره سازی ۲ با موفقیت بوت می شود

T4: دستگاه ذخیره سازی ۲ در مرحله بوت دچار مشکل شده و بوت نمی شود.

T5: سیستم میزبان ۱ با موفقیت بوت می شود

T6: سیستم میزبان ۱ در مرحله بوت دچار مشکل شده و بوت نمی شود.

T7: سیستم میزبان ۲ با موفقیت بوت می شود

T8: سیستم میزبان ۲ در مرحله بوت دچار مشکل شده و بوت نمی شود.

T9: همه مولفه های اصلی سیستم (دستگاه های ذخیره سازی و سیستم های میزبان) به درستی

بوت شده اند.

T10 و T11: دو دستگاه ذخیره سازی به درستی بوت شده اند در حالی که تنها یکی از دو

سیستم میزبان به سلامت مرحله بوت را گذرانده و دیگری دچار مشکل شده است.

T12 و T13: دو سیستم میزبان به درستی بوت شده اند در حالی که تنها یکی از دو سیستم

ذخیره سازی به درستی مرحله بوت را پشت سر گذاشته است.

T14، T15، T16 و T17: تنها یکی از دستگاه های ذخیره سازی و یکی از سیستم های میزبان

به درستی مرحله بوت را پشت سر گذاشته اند و دیگری در این مرحله ناکام مانده است.

T18، T19: تنها یکی از دو دستگاه ذخیره سازی بوت شده است و دیگری به همراه هر دو

سیستم میزبان دچار مشکل شده اند.

T20، T21: تنها یکی از دو سیستم میزبان بوت شده است و دیگری به همراه هر دو دستگاه

ذخیره سازی دچار مشکل شده اند.

- T22: اعلام آمادگی برای ورود به مرحله بعد با همه توان پردازشی ممکن
- T23: اعلام آمادگی برای ورود به مرحله بعد با هر دو سیستم ذخیره سازی و تنها یک میزبان
- T24: اعلام آمادگی برای ورود به مرحله بعد با یک سیستم ذخیره سازی و هر دو سیستم میزبان
- T25: اعلام آمادگی برای ورود به مرحله بعد با تنها یک سیستم ذخیره سازی و یک میزبان
- T26: تلاش برای رفع مشکل نرم افزاری در مولفه های ناکارا
- T27: ایجاد ماشین های مجازی لازم بر روی ESX ها و روشن کردن آنها
- T28: ایجاد یک HA cluster و اجرا کردن سرویس HA بر روی تعدادی از VM ها
- T29: اعلام آمادگی سیستم برای ورود به مرحله بعد
- T30: سرویس های سطح بالای دیگری مورد نیاز است.
- T31: ایجاد یک جفت fault tolerance
- T32: سرویس های سطح بالای دیگری مورد نیاز است.
- T33: اعلام آمادگی سیستم برای ورود به مرحله بعد
- T34: اعلام آمادگی سیستم برای ورود به مرحله بعد بدون امکان اجرای سرویس های سطح بالا.
- این حالت به دلیل در دسترس نبودن تعداد کافی مولفه های سخت افزاری بروز نموده است.
- T35: ایجاد توازن و بهینگی در میزان منابع تخصیص یافته به ماشین های مجازی و سرویس ها و نیز رزرو کردن منابع لازم برای موارد خاص از جمله بروز مشکل در یکی از سیستم های میزبان و غیره. این کار توسط سرویس DRS صورت می گیرد.
- T36: ESX درخواست ماشین های مجازی در حال اجرا را دریافت و به آنها سرویس می دهد.
- T37: ارائه بازخورد سرویس دهی به سیستم
- T38: پردازش داخلی سرویس FT برای انتقال کنترل از یک ماشین مجازی به جفت آن در شرایط بروز مشکل برای ماشین مجازی اول
- T39: ارائه بازخورد سرویس دهی به سیستم
- T40: طرح درخواست مهاجرت یک ماشین مجازی توسط مدیر سیستم. این درخواست توسط سرویس VMotion اجرا خواهد شد.

- T41: پردازش داخلی سرویس VMotion برای انتقال یک ماشین مجازی از یک سیستم میزبان به میزبان دیگر
- T42: ارائه بازخورد سرویس Motion به سیستم
- T43: پردازش داخلی کلاستر HA
- T44: ارائه بازخورد سرویس HA به سیستم
- T45: هر دو دستگاه ذخیره سازی به درستی بوت شده اند و در عین حال، هر دو سیستم میزبان در این مرحله دچار مشکل شده اند.
- T46: هر دو سیستم میزبان به درستی بوت شده اند و در عین حال، هر دو دستگاه ذخیره سازی در این مرحله دچار مشکل شده اند.
- T47: تمام چهار مولفه اصلی سیستم در مرحله بوت دچار مشکل شده اند.
- T48: مسئول رفع اشکال، موفق به برطرف کردن مشکل مولفه های نقص یافته می شود.
- T49: مسئول موفق به رفع ایراد نمی شود.
- T50: بارگزاری ماشین مجازی دچار اختلال می شود
- T51: ایجاد کلاستر HA دچار اختلال شده و ناکام مانده است
- T52: ایجاد جفت FT دچار اختلال شده و ناکام مانده است
- T53: بروز اختلال در سرویس دهی ESX به ماشین های مجازی
- T54: بروز اختلال در سرویس FT
- T55: بروز اختلال در سرویس VMotion
- T56: بروز اختلال در سرویس HA
- T57: ارسال بازخورد در مورد عدم موفقیت در ایرادیابی فیزیکی و بازگشت به وضعیت اولیه
- T58: رفع اشکال در بارگزاری ماشین مجازی
- T59: رفع اشکال در ایجاد کلاستر HA
- T60: رفع اشکال در ایجاد جفت FT
- T61: برطرف کردن اشکال در روند سرویس دهی ESX

T62: برطرف کردن اشکال در روند سرویس دهی FT

T63: برطرف کردن اشکال در روند سرویس دهی VMotion

T64: برطرف کردن اشکال در روند سرویس دهی FT

پس از شرح مختصر گزارها در مدل که باعث تغییر وضعیت آن و به وجود آمدن توزیع های مختلف توکن در مدل می شوند، به تحلیل دقیقتر آن می پردازیم. در این بخش برای تحلیل بهتر مدل می توانیم آن را به چند مرحله^۱ تقسیم نمائیم تا بتوانیم رفتار سیستم را گروه بندی کنیم. پس از شناخت رفتار این دیتا سنتر نمونه، می توان رفتار آن را از جنبه های مختلف به کمک ابزار فرمال ارزیابی و تحلیل نمود.

به این منظور، انتقال کنترل در مدل را به مراحل زیر تقسیم می کنیم:

مرحله ۱: از زمان روشن کردن مولفه های سخت افزاری شامل دستگاه های ذخیره سازی و سیستم های میزبان تا انتهای زمان بوت شدن یا دچار مشکل شدن آن ها (P0-P17 و P35، P36).

مرحله ۲: از هنگام آمادگی سیستم برای ایجاد و بارگزاری ماشین های مجازی تا هنگام آمادگی سیستم برای ایجاد توازن و بهینگی در میزان منابع تخصیص یافته به ماشین های مجازی و سرویس ها توسط سرویس DRS (P18-P23 و P37-P39).

مرحله ۳: اجرا شدن سرویس DRS و ایجاد تعادل و توازن در میزان منابع تخصیص یافته به ماشین های مجازی و سرویس های در حال اجرا (P24)

مرحله ۴: واگذار کردن کنترل به ESX در حال اجرا بر روی سیستم های میزبان و نیز سرویس های سطح بالا مانند HA cluster ها و FT pair ها برای کنترل و مدیریت دینامیک سیستم (-P25 P33 و P40-P43).

با این تقسیم بندی، در نگاه اول می توان دریافت که نحوه انتقال کنترل در مراحل ۲ الی ۴ به شدت وابسته به نحوه اجرای کار در مرحله ۱ است. با این توضیح که با توجه به تعداد سیستم های ذخیره سازی و تعداد سیستم های میزبان بوت شده، می توان تعیین کرد که آیا سیستم قابلیت اجرای سرویس های سطح بالا را دارد یا خیر. به عنوان مثال، یک جفت fault tolerance برای

¹ Stage

اجرای صحیح خود نیاز به حداقل دو سیستم میزبان دارد. همچنین میزان منابع تخصیص یافته به ماشین های مجازی و نیز منابع رزرو شده کاملاً وابسته به همین نکته است. در مرحله آخر نیز با توجه به سرویس های اجرا شده و کلاسترها و جفت های به وجود آمده، گردش کار در سیستم رخ می دهد. در انتها مدیر سیستم نیز با توجه به تعداد سیستم های میزبان در دسترس می تواند سرویس هایی مانند مهاجرت ماشین های مجازی توسط VMotion را درخواست نماید. با توجه به اینکه در این سناریوی ساده تنها دو سیستم ذخیره سازی و دو میزبان آورده شده است، می توان دریافت که در یک دیتا سنتر واقعی میزان پیچیدگی فضای حالت بسیار بیش از این خواهد بود. گزاره های T9 الی T21 و T45 الی T47 حالاتی را که ممکن است در مورد بوت شدن یا مختل شدن سیستم ها رخ دهد نشان می دهد. همانطور که دیده می شود، یک جایگشت بین حالات ممکن رخ داده است و در نهایت، این حالات به پنج گروه کلی تقسیم شده اند:

۱- اگر تمام چهار عنصر به درستی بوت شوند سیستم در حالت P13 یا بوت کامل^۱ قرار می گیرد. این حالت که انتظار مدیر از سیستم است امکان اجرا کردن تمامی سرویس های سطح بالا و حداکثر تعداد ماشین های مجازی که سیستم می تواند داشته باشد را فراهم می کند.

۲- اگر هر دو سیستم ذخیره سازی بوت شود اما تنها یکی از سیستم های میزبان به درستی بوت شود و دیگری مختل گردد، سیستم در وضعیت P14 قرار می گیرد. در این حالت امکان ایجاد و راه اندازی ماشین های مجازی به تناسب منابع پردازشی میزبان بوت شده وجود دارد اما به دلیل ناکافی بودن تعداد سیستم های میزبان، سرویس هایی مانند HA و fault tolerance اجرا نخواهد شد.

۳- در صورتی که هر دو سیستم میزبان و نیز یکی از سیستم های ذخیره سازی به درستی بوت شوند، سیستم در حالت P15 قرار می گیرد. که در این حالت نیز امکان اجرا شدن سرویس ها و نیز ماشین های مجازی وجود خواهد داشت. منتها با توجه به اینکه در این معماری هر ماشین مجازی در واقع یک فایل بر روی دیسک می باشد، امکان ارائه برخی

^۱ Perfect boot up

از سرویس ها محدود خواهد شد. در مورد جزئیات این سرویس ها و مدل فرمال آنها در بخش های بعدی به تفصیل صحبت خواهد شد.

۴- اگر تنها یک سیستم ذخیره سازی و یک میزبان به درستی بوت شوند، سیستم در وضعیت P16 قرار می گیرد. این حالت، حداقلی است که سیستم برای ادامه کار خود و ارائه سرویس به آن نیاز دارد. به این معنی که با حداقل یک سیستم ذخیره سازی و یک سیستم میزبان، یک دیتا سنتر می تواند ماشین های مجازی را ایجاد و اجرا نماید. طبیعتاً در این حالت امکان ارائه سرویس های سطح بالا نیز وجود نخواهد داشت.

۵- و در نهایت، اگر تنها یکی از چهار مولفه نام برده شده بوت شود و یا هیچ کدام بوت نشوند، سیستم در وضعیت P17 قرار می گیرد. در این حالت گفته می شود سیستم دچار اختلال شده است و امکان ادامه مراحل وجود ندارد. اگر تنها یک سیستم ذخیره سازی بدون هیچ سیستم میزبانی بوت شده باشد، منبع پردازشی برای اجرای ماشین های مجازی وجود ندارد. همچنین اگر تنها یکی از سیستم های میزبان بدون هیچ منبع ذخیره سازی بوت شده باشد، امکان ذخیره سازی و بازیابی فایل های مربوط به ماشین های مجازی و سیستم عامل آنها وجود ندارد پس باز هم سیستم دچار بن بست خواهد شد. در این وضعیت باید تلاش شود تا سیستم طی چند مرحله ایراد یابی و دوباره بوت شود.

در صورت بروز هریک از حالات ۱ تا ۴، سیستم پس از ایجاد و اجرای ماشین های مجازی، بر اساس تعداد و نوع مولفه های بوت شده توانایی اجرای سرویس های سطح بالا را خواهد داشت. در صورت فعال شدن P19، با این معنی که سیستم توانایی ارائه سرویس های سطح بالا را دارد، بر حسب اینکه مدیر سیستم مایل به تعریف کلاستر HA و یا جفت های FT می باشد، سیستم می تواند گام های متناظر را طی نماید. این گام ها برای سرویس اول (HA) گزارهای T28 و T30 و برای سرویس دوم (FT) گزارهای T31 الی T32 خواهد بود. با هر بار اجرا شدن این حلقه ها، یک کلاستر HA یا یک جفت FT تعریف می گردد. بنابراین برای تعریف چندین نمونه از آنها حلقه ها باید چند بار طی شوند. هرگاه که یک کلاستر HA تعریف می شود یا یک ماشین مجازی به یک کلاستر HA اضافه می گردد، سیستم در وضعیت P21 قرار می گیرد. همچنین هرگاه یک جفت

FT تعریف می شود سیستم در وضعیت P22 قرار می گیرد. قرار گرفتن در این دو حالت به این معنی است که مدیر می تواند سیستم را به وضعیت های P18 و P19 برای تعریف نمونه جدیدی از سرویس ها برگرداند و یا می تواند به تعریف سرویس ها خاتمه داده و سیستم به مرحله بعدی برود. این بدان معنی است که هر دو وضعیت "آماده برای مرحله بعد" (گزاره های T29 و T33) و نیز "بازگشت برای تعریف نمونه جدیدی از سرویس ها" (گزاره های T30 و T32) در این مرحله فعال^۱ هستند اما اجرا شدن گزاره ها وابسته به تصمیم سیستم خواهد بود.

بعد از مراحل گفته شده، در مرحله سوم که مرحله کوچک اما بسیار حیاتی برای سیستم می باشد، منابع سخت افزاری بر روی سیستم های میزبان زمان بندی و متعادل^۲ می شوند. این فرایند به وسیله سرویس هوشمند DRS صورت می پذیرد که در آن بر اساس ماشین های مجازی اجرا شده و نیز سرویس های سطح بالای راه اندازی شده، مقدار متناسبی از منابع پردازشی و حافظه به ماشین ها تخصیص یافته و مقداری نیز به اجرای سرویس ها واگذار می شود. همچنین درصد مناسبی از منابع برای اجرا در مواقع ضروری مانند HA و VMotion رزرو می شود. بدون اجرا شدن موفقیت آمیز این مرحله، ما اطمینانی از توانایی سیستم در تخصیص منابع لازم به ماشین های مجازی در زمان بروز مشکل در سیستم های میزبان یا سیستم های ذخیره سازی نخواهیم داشت. نحوه کار این سرویس ها در بخش های بعدی تشریح و تحلیل خواهد شد.

در مدل مذکور، اجرای سرویس DRS در گزار T35 نشان داده شده است. همانطور که مشاهده می شود این گزار یک گزار زمان دار^۳ است که به معنی اجرای تعداد زیادی از زیرپروژه ها برای اجرای آن می باشد.

آخرین نکته قابل توجه در مرحله سوم این است که اگر بر مبنای خروجی مرحله دوم، امکان ارائه سرویس های سطح بالا به دلیل کمبود منابع سخت افزاری وجود نداشته باشد، سیستم در حالت

¹ Enable

² Fire

³ Balance

⁴ Timed transition

P23 قرار خواهد گرفت. در این وضعیت مدیر سیستم تنها قادر به راه اندازی ماشین های مجازی بر حسب میزان منابع سیستم خواهد بود و سرویس ها ارائه نخواهند شد. در این شرایط، تنها پس از اجرای موفقیت آمیز ماشین های مجازی، گزار T34 فعال می گردد که با اجرا شدن آن سیستم برای اجرای سرویس DRS آماده خواهد بود.

مرحله چهارم که مرحله نهایی نیز می باشد، مربوط به قرار گرفتن سیستم در وضعیت طبیعی اجرایی و انتقال کنترل هر ماشین میزبان به ESX آن است. در این مرحله که کنترل سیستم را برای همیشه در دست خواهد داشت، هر ESX مسئول سرویس دادن به ماشین های مجازی اجرا شده بر روی خود است. همچنین در هر زمانی به دلیل شرایط محیطی یا به درخواست مدیر سیستم، هر یک از سرویس های سطح بالا می تواند اجرا شود. این سرویس ها به کمک سیگنال های ضربانی^۱ یا مکانیزم های مبتنی بر agent از شرایط محیطی که باید در آن فعال شوند مطلع می گردند. لازم به ذکر است که پروسه های داخلی هر یک از گزارهای T36، T38، T41 و T43 مدل های پیچیده ای برای خود دارند. به عبارت دیگر در یک مدل سلسله مراتبی، ما این پروسه را در یک گزار خلاصه کرده ایم اما در بخش های بعدی با دقیقت بیشتر شدن بر روی نحوه کار آنها، مدل هایی برای این بخش ها ارائه خواهیم نمود.

۴.۲.۱. ارزیابی و تحلیل مدل فرمال

در این بخش با تحلیل مدل ارائه شده در بخش قبل، سعی خواهیم کرد حقایقی را در مورد سیستم استخراج نمائیم. با یادآوری از فصل ۲، در این بخش به بررسی خصوصیات رفتاری سیستم در مدل شبکه پتری ارائه شده می پردازیم تا خصوصیات Liveness، Safeness و Reversibility را بر روی آن تحلیل نمائیم. در نهایت میزان خوش رفتاری سیستم را در فضای حالت مورد بررسی تعیین می نمائیم. در ادامه قضایایی برای تحلیل مدل ارائه شده اند.

¹ Heartbeat mechanism

Liveness ۴,۲,۱,۱

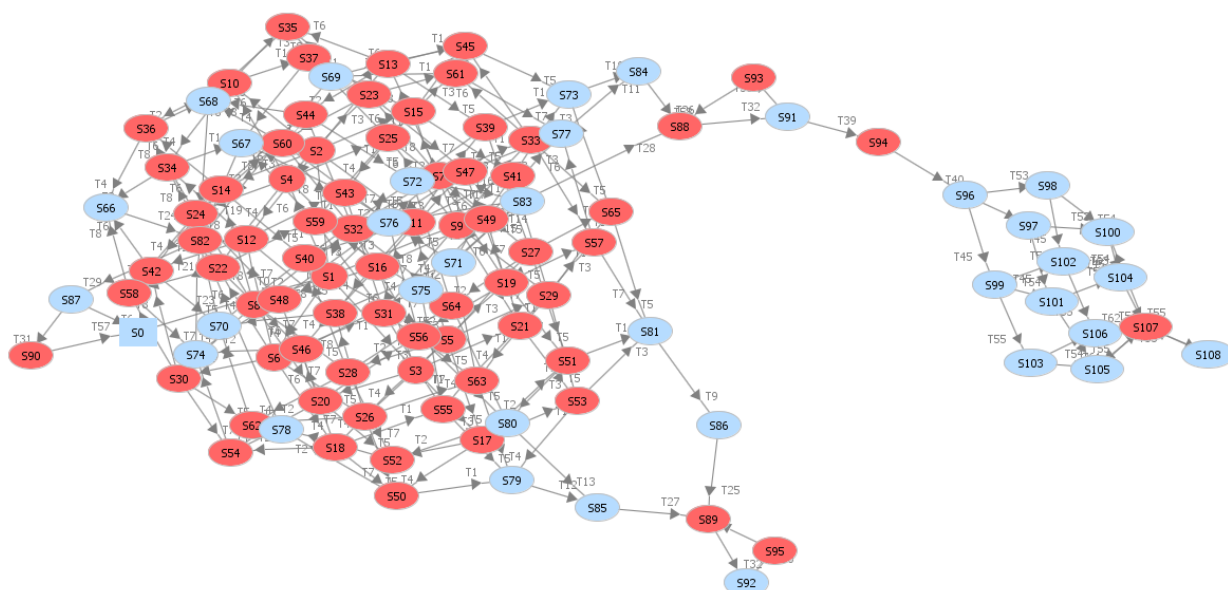
با تکیه به قضیه ۱ از فصل ۲ و توجه به مدل ارائه شده، واضح است که هیچ موقعیت یا گزار source و یا sink در آن وجود ندارد بنابر این با این تعریف اولیه، سیستم دارای خصوصیات Liveness و Safeness می باشد. با این حال با ارائه دلایل بیشتر مدل را بررسی می کنیم.

Safeness ۴,۲,۱,۲

بر اساس تعریف درخت پوشا، می توان قضیه ۲ را به گراف پوشا نیز تعمیم داد. در این صورت با بررسی گراف پوشای این مدل می توانیم از safe بودن سیستم مطمئن شویم.

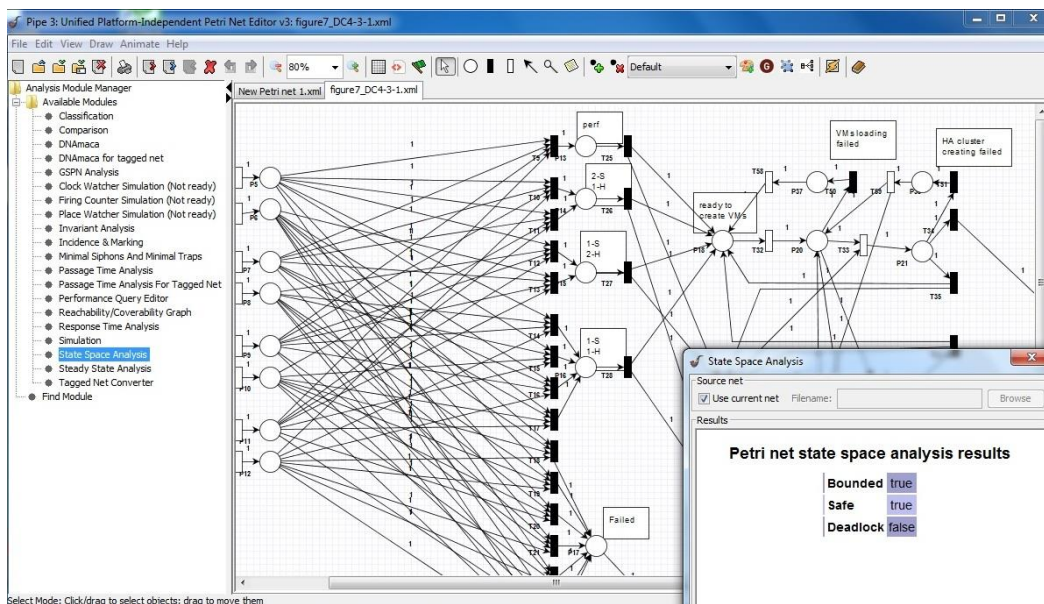
در این تحقیق برای تحلیل مدل های طراحی شده از نرم افزار بررسی مدل شبکه پتری PIPE^۱ نسخه ۳ استفاده شده است. گراف پوشای بدست آمده برای مدل ۴,۴ به کمک این نرم افزار در شکل ۴,۵ آمده است. با بررسی این گراف در میابیم که تمام لبه ها دارای برچسب ۰ و ۱ بوده اند. این بدان معنی است که در تمام تغییر حالات از M_i به M_j تعداد توکن های موجود در موقعیت ها هیچ گاه بیشتر از ۱ نبوده است. با این استدلال می توان مطمئن بود که سیستم 1-bounded و در نتیجه safe می باشد.

¹ Platform independent Petri net Editor



شکل ۴.۵ گراف پوشا برای مدل پتری شکل ۴.۴

برای اطمینان از درست بودن نتیجه به دست آمده، خصوصیات رفتاری شبکه پتری مذکور را با نرم افزار PIPE بررسی می کنیم. حاصل این بررسی نشان می دهد که مدل دارای خصوصیات Boundedness و Safeness بوده و دارای deadlock نیز نمی باشد به این معنی که سیستم Live است (شکل ۴.۶).



شکل ۴,۶. نتیجه تحلیل فضای حالت به وسیله نرم افزار PIPE

۴,۲,۱,۳. Reversibility

برای بررسی این خصوصیت لازم است به تعریف Reversibility توجه کنیم. با یاد آوری تعریف این مفهوم و نیز خاصیت Reachability از فصل ۲ داریم: شبکه پتری (N, M_0) Reversible، خواهنده می شود اگر برای هر وضعیت M_1 ، در دنباله قابل اجرا از M_0 ، $(R(M_0))$ از طریق M ، reachable باشد.

با این تعریف، در شبکه های کوچک می توان با بررسی کلیه وضعیت های محتمل در شبکه، reachable بودن M_0 را از طریق آنها بررسی نمود. در صورتی که M_0 از طریق تک تک این وضعیت ها (M_1 تا M_n) قابل دسترسی باشد این شبکه Reversible است. با توجه به اینکه این وضعیت ها تماماً در گراف پوشای مدل منعکس می شوند می توان با بررسی گره های گراف این حالات را بررسی نمود.

به این ترتیب در صورت پیدا کردن یک مثال نقض در گراف ۴,۵، می توان از عدم وجود خاصیت Reversibility در مدل اطمینان یافت. با بررسی گراف می بینیم که با قرار گرفتن توکن در

¹ Marking

موقعیت P25 و اجرای T40 (وضعیت S94)، سیستم به وضعیت S96 رفته و از این به بعد امکان بازگشت به S0 (M_0) وجود ندارد.

در مثال دیگری با اجرا شدن T25 یا T27 سیستم در وضعیت S89 قرار می گیرد. در این حالت در موقعیت های P18 و P19 توکن وجود دارد. با بررسی گراف می بینیم که اینجا نیز امکان بازگشت سیستم به حالت اولیه (M_0) وجود ندارد.

به این ترتیب ملاحظه می شود که مدل ۴,۴ دارای خاصیت Reversibility نمی باشد. با بررسی سیستم واقعی خواهیم دید که مدل متناظر نباید به طور مطلق Reversible باشد. زیرا پس از بوت اولیه سیستم و ایجاد و راه اندازی ماشین های مجازی و سرویس های سطح بالا، کنترل سیستم به مرحله ۴ منتقل شده و سیستم در وضعیت نرمال اجرای درخواست های ماشین های مجازی توسط ESX و اجرای سرویس های سطح بالا در صورت لزوم باقی می ماند و هیچ گاه به مرحله نخست راه اندازی باز نمی گردد. با این حال می توان گفت که سیستم به طور ضمنی (جزئی) Reversible است زیرا پس از انتقال کنترل به مرحله ۴ (موقعیت های P25, P27, P29 و P32) توکن ها به صورت گردشی در این بخش حرکت کرده و پس از هربار اجرای یک سرویس به موقعیت اولیه خود در این مرحله باز می گردند.

بنابراین با توجه به برآورده نشدن خصوصیت Reversibility به طور مطلق، سیستم به طور خوش رفتار نمی باشد و به آن نیازی نیز ندارد.

۴,۲,۲. بررسی لایه های پایینتر مدل فرمال

در این بخش از تحقیق لازم به ذکر است که این مدل انتزاعی ترین سطح تصور ما از دیتا سنتر طرح شده در بخش گذشته است. با این توضیح که بخش های زیادی از وظایف لازم اجرا در چنین سیستمی در این مدل از نظر دور مانده و تنها به طرح یک گزار برای نشان داده کلیه وظایف فوق بسنده شده است. این کار به منظور ارائه یک مدل کلی برای بررسی رفتار کلی دیتا سنتر انجام شده است و به معنی حذف بخش های مذکور نمی باشد. در مدل شکل ۴,۴، کلیه گزارهایی که به شکل توخالی نشان داده شده اند نشان دهنده مجموعه ای از وظایف و مسئولیت های دیتا سنتر هستند

که برای ایجاد انتزاع بیشتر خلاصه شده است. به عنوان مثال گزار T28 نشان دهنده کلیه اعمال لازم برای ایجاد یک کلاستر HA یا اضافه کردن یک سیستم میزبان به یک کلاستر موجود است. یا گزار T36 بیانگر پروسه سرویس دهی ESX به ماشین های مجازی خود در صورت صدور درخواست از جانب آنها است.

به این ترتیب و برای بررسی دقیق و جامع تر دیتا سنتر نمونه، لازم است مدلی برای هر یک از زیر مجموعه های این طرح کلی طراحی شده و آنها را نیز ارزیابی و تحلیل نمائیم. به این ترتیب می توان اطمینان بیشتری نسبت به نحوه کار سیستم و مشخصاتی از جمله پایداری، توسعه پذیری و انعطاف پذیری آن داشت.

۴,۳. بررسی نحوه کار سرویس HA

در این بخش ابتدا به شرح نحوه کار سرویس VMware HA می پردازیم و پس از آن مدل شبکه پتری مربوط به این سرویس طراحی و تحلیل خواهد شد.

۴,۳,۱. شرح سرویس VMware HA

به طور کلی می توان گفت سرویس HA در حال کار بودن دائمی ماشین های مجازی را تضمین می کند. با این توضیح که این سرویس سیستم های میزبان را که از طرف مدیر سیستم مشخص شده در یک کلاستر قرار می دهد و به طور دائمی آنها را مانیتور می کند. هر گاه مشکلی برای میزبان مربوطه پیش بیاید و به هر دلیل از مدار خارج شود، این سرویس ماشین های مجازی در حال اجرا بر روی سیستم میزبان مختل شده را به سرعت بر روی سیستم های دیگر راه اندازی می کند [۶۱].

¹ Failure events

مفهوم میزبانان اصلی^۱ و میزبانان ثانویه^۲

وقتی مدیر سیستم در زمان راه اندازی اولیه دیتا سنتر و یا پس از آن در هنگام کار آن میزبانی را به یک کلاستر HA اضافه می کند، یک agent بر روی میزبان نصب می شود که وظیفه برقراری ارتباط با بقیه میزبانان حاضر در آن کلاستر را بر عهده دارد. به طور قراردادی، پنج میزبان اول در کلاستر به عنوان میزبانان اصلی و بقیه میزبانان افزوده شده به کلاستر به عنوان میزبانان ثانویه معین می شوند. میزبانان اصلی وظیفه اطلاع رسانی در مورد وضعیت کلاستر به همه میزبانان را بر عهده دارند. همچنین در صورت بروز مشکل و اختلال برای یکی از اعضای کلاستر، این میزبانان اصلی هستند که پروسه failover را شروع و سازمان دهی می کنند. اگر یکی از میزبانان اصلی به هر دلیل از مدار خارج شود (مثلا به دلیل بروز اختلال و یا خاموش شدن عمدی با هدف تعمیر و نگهداری) سرویس HA یکی از میزبانان ثانویه را به عنوان میزبان اصلی منسوب می کند.

هر میزبانی که عضو یک کلاستر HA می شود باید لزوماً با یکی از میزبانان اصلی ارتباط برقرار کند تا پیکربندی سرویس HA بر روی آن تکمیل گردد. این وضعیت در مورد اولین میزبانی که به یک کلاستر اضافه می شود استثنا است. باید توجه داشت که وجود حداقل یک میزبان اصلی در یک کلاستر برای کارکرد درست آن ضروری است. به این معنی که اگر همه میزبانان اصلی غیر قابل دسترسی شوند و قادر به پاسخگویی نباشند، هیچ میزبان دیگری نمی تواند عضو کلاستر مذکور شود. یکی از سیستم های میزبان اصلی به عنوان میزبان اصلی فعال^۳ معین می شود و وظایف زیر را بر عهده خواهد داشت:

- تصمیم گیری در مورد اینکه ماشین های مجازی مربوط به یک میزبان مختل شده در کجا و روی کدام میزبان باید دوباره راه اندازی شوند
- به خاطر نگه داشتن وضعیت ماشین های مجازی و میزبانانی که در مرحله راه اندازی مجدد (restart) دچار اختلال شده اند.

¹ Primary Hosts

² Secondary Hosts

³ Active primary host

- تعیین اینکه چه زمانی برای تلاش مجدد برای restart یک ماشین مجازی مناسب است.

اگر یک میزبان اصلی فعال دچار مشکل شده و از مدار خارج شود، یک میزبان اصلی دیگر جای آن را می گیرد.

در این بخش با دقیقتر شدن روی مدل ارائه شده در بخش قبل، به مدل سازی و تحلیل نحوه کار سرویس HA در دیتا سنتر نمونه خواهیم پرداخت. این مدل در واقع شرح نحوه انتقال کنترل در مجموعه وظایف خلاصه شده در گزارهای بین موقعیت های P43 و P44 از مدل شکل ۴,۴ است. به عبارت دیگر، P0 در این مدل معادل P43 در مدل ۴,۴ می باشد.

تشخیص خطا و ایزوله شدن میزبان از شبکه

Agent ها با یکدیگر ارتباط برقرار می کنند و به این ترتیب از فعال و سالم بودن میزبانان عضو کلاستر مطمئن می شوند. این فرایند با تبادل سیگنال های ضربانی^۱ که به طور پیش فرض هر ۱ ثانیه یک بار انجام می شود صورت می پذیرد. اگر با گذشت یک بازه زمانی ۱۵ ثانیه ای هیچ سیگنالی از طرف یک میزبان دریافت نگردد، و پس از آن میزبان مذکور به درخواست ping جواب ندهد، این میزبان یک کامپیوتر از کار افتاده و از مدار خارج شده محسوب می شود. در صورت از کار افتادن یک میزبان، کلاستر تلاش می کند تا ماشین های مجازی در حال کار روی آن میزبان را رفع ایراد نماید. به این معنی که این ماشین ها را بر روی میزبانان دیگری که منابع آزاد یا رزرو شده دارند restart نماید. همانطور که در فصل قبلی اشاره شد منظور از منابع در اینجا به طور اعم پردازنده و حافظه می باشد.

ایزوله شدن یک میزبان از شبکه هنگامی رخ می دهد که میزبان کاملاً سالم و در حال کار است اما نمی تواند با هیچ یک از میزبانان دیگر در کلاستر ارتباط برقرار کند و سیگنال ضربانی را مبادله نماید. به طور پیش فرض اگر میزبانی به مدت ۱۲ ثانیه سیگنالی از هیچ یک از میزبانان حاضر در

¹ Heartbeat exchange

² Failover

کلاستر دریافت نکند، تلاش می کند تا آنها را ping نماید. در صورت عدم موفقیت در دریافت جواب ping (ping reply)، خود را به عنوان یک میزبان ایزوله شده از شبکه شناسایی می کند.

در صورتی که میزبان ایزوله شده نتواند ارتباط خود با شبکه و بقیه میزبانان کلاستر را در طول یک بازه زمانی ۱۵ ثانیه ای دوباره برقرار کند، میزبانان دیگر کلاستر آن را به عنوان یک میزبان از کار افتاده تلقی کرده و تلاش خواهند کرد تا ماشین های مجازی روی آن را رفع ایراد کنند. اما با توجه به اینکه میزبان ایزوله شده هنوز فعال است و به منبع ذخیره سازی مشترک دسترسی دارد، این میزبان هنوز کنترل فایل های مربوط به ماشین های مجازی خود را در اختیار دارد و این فایل ها برای میزبان مذکور قفل هستند. لازم به توضیح است که برای جلوگیری از تخریب داده های مربوط به ماشین های مجازی و فایل های آنها، VMFS مکانیزمی برای جلوگیری از دسترسی همزمان چندین میزبان به یک ماشین مجازی دارد که توسط آن، فایل های مربوط به یک ماشین مجازی در حال کار را قفل می کند تا میزبانان دیگر نتوانند عملیات خواندن و نوشتن انجام دهند.^۱ بنابراین تلاش برای راه اندازی مجدد ماشین های مجازی در حال کار بر روی یک میزبان ایزوله شده با شکست روبرو خواهد شد. برای رفع این مشکل، میزبان ایزوله شده پس از اطمینان از وضعیت قطعی خود از شبکه، اقدام به خاموش کردن ماشین های مجازی خود می کند. به این ترتیب قفل مربوط به فایل ماشین های مذکور آزاد شده و آنها می توانند بر روی میزبانان در حال کار دیگر در کلاستر راه اندازی شوند.

استفاده همزمان VMware HA و DRS

استفاده همزمان سرویس HA و سرویس زمان بندی تخصیص منابع (DRS) در یک دیتا سنتر، امکان ترکیب رفع اشکال اتوماتیک ماشین های مجازی را به همراه ایجاد تعادل در بار اجرایی بر روی منابع آفرام می نماید. ترکیب درست این دو سرویس به ما امکان متعادل کردن سریع تر بار وارد

^۱ VMFS disk lock

^۲ Load balancing

شده از سوی ماشین های مجازی به میزبانان، بعد از انتقال آنها توسط سرویس HA از یک میزبان به میزبان دیگر را می دهد.

بعد از اینکه سرویس HA فرایند رفع اشکال یک میزبان از کار افتاده را انجام داد و ماشین های مجازی وی را بر روی میزبانان سالم کلاستر راه اندازی نمود، اولین اولویت اجرایی آن، فراهم کردن امکان دسترسی به همه ماشین های مجازی مذکور^۱ خواهد بود. پس از اینکه ماشین های مجازی روی میزبانان جدید مجدداً راه اندازی شد، میزبانان جدید این ماشین ها ممکن است به شدت تحت فشار کمبود منابع باشند در حالی که میزبانان دیگری در کلاستر بار پردازشی کمی دارند. باید توجه داشت که سرویس HA با توجه به میزان پردازنده و حافظه رزرو شده اقدام به انتقال ماشین ها می کند ولی میزان مصرف واقعی منابع توسط ماشین های مجازی ممکن است با این تخمین متفاوت باشد.

همچنین ذکر این نکته ضروری است که در فرایند انتقال ماشین های مجازی به وسیله سرویس HA و برقراری تعادل در منابع مصرفی میزبانان توسط سرویس DRS، اعمال گفته شده در مورد میزبانانی که در وضعیت نگهداری^۲ هستند انجام نخواهد شد. به عبارت دیگر ماشین های مجازی از روی این میزبانان تخلیه نمی شوند. این امر به دلیل حفظ منابع رزرو شده برای مرحله رفع اشکال انجام می شود. در صورت نیاز به انتقال این ماشین های مجازی، مدیر سیستم باید این کار را به صورت دستی و به کمک سرویس VMotion انجام دهد.

در ادامه این بخش به طراحی یک مدل فرمال و تحلیل رفتار های سرویس VMware HA بر روی مدل مربوطه خواهیم پرداخت.

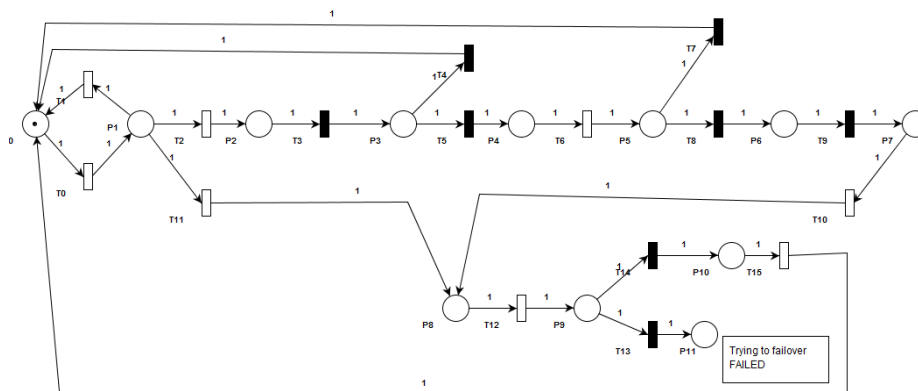
۴.۳.۲. ارائه مدل فرمال از نحوه کار سرویس HA

در این قسمت به ارائه مدلی فرمال از سرویس VMware HA به کمک ابزار شبکه های پتری می پردازیم تا بتوانیم تصویر دقیقتری از نقاط ضعف و نحوه کار آن به دست آوریم.

¹ Availability

² Maintenance

با توجه به نحوه کار شرح داده شده در قسمت قبل، و با این فرض که در دیتاستر مربوطه این سرویس همراه با DRS کار می کند، می توان مدل زیر را برای آن ارائه داد.



شکل ۴،۷. مدل پتری نحوه کار سرویس HA

در ادامه به شرح گزارها و مراحل تغییر حالت در سیستم فوق می پردازیم.

T0: سیگنال ضربانی بر روی سیستم های میزبان عضو کلاستر تولید می شود.

T1: سیگنال تولید شده بین سیستم های عضو یک کلاستر HA تبادل می شود

T2: دریافت سیگنال ضربانی به مدل ۱۲ ثانیه از همه سیستم ها قطع می شود.

T3: سیستم میزبان اقدام به ping کردن سیستم های دیگر می کند.

T4: پیام ping reply از سیستم های دیگر دریافت می شود که به معنی برقراری مجدد ارتباط و از بین رفتن اختلال است.

T5: درخواست ping با شکست روبرو می شود که به معنی باقی بودن اختلال است.

T6: سیستم میزبان به مدت ۱۵ ثانیه برای برقراری مجدد ارتباط و دریافت پیام ping reply تلاش می کند.

T7: سیگنال ضربانی دریافت می شود.

T8: در برقراری مجدد ارتباط با دیگر سیستم های کلاستر دچار مشکل می شود.

T9: در این شرایط، با توجه به قطع شدن ارتباط با همه سیستم های میزبان، سیستم مذکور خود را به عنوان یک سیستم ایزوله شناسایی می کند. به این معنی که می فهمد ارتباط خودش با بقیه قطع شده است.

T10: سیستم میزبان ماشین های مجازی خود را خاموش می کند.

T11: ارتباط سیستم میزبان با یکی از سیستم های عضو کلاستر به مدت ۱۵ ثانیه قطع می شود.

T12: کلاستر تلاش می کند تا سیستم ایزوله شده یا از کار افتاده را ایرادیابی نماید. به این معنی که ماشین های مجازی که روی آن در حال کار بوده اند را بر روی میزبانان دیگر که منابع خالی یا رزرو شده دارند راه اندازی می کند.

T13: فرایند ایرادیابی دچار اختلال شده و عقیم می ماند.

T14: فرایند ایرادیابی تکمیل شده و سیگنال ضربانی دوباره تبادل می شود.

T15: در صورتی که سیستم میزبان از کار افتاده یک میزبان اصلی بوده باشد، برای جبران کمبود

تعداد سیستم های اصلی، یکی از سیستم های میزبان ثانویه به میزبان اصلی تبدیل می شود.

با توجه به اینکه مدل فوق زیر مجموعه ای از مدل ۴,۴ می باشد، برخی از موقیعت ها و گزارها در این مدل با مدل پدر در تناظر هستند. از آن جمله می توان به P0 (معادل P43 در مدل ۴,۴) و T13 (معادل T56 در مدل ۴,۴) اشاره نمود. به این ترتیب اگر مدل ۴,۴ را یک مدل لایه ۱ در نظر بگیریم، طرح جزئیات نحوه کار سرویس HA یک مدل در لایه ۲ به دست خواهد داد که به کمک آن می توانیم جزئیات بیشتری از رفتار سیستم را بررسی نمائیم.

همانگونه که در این مدل مشخص شده است، هیچ راهکاری برای برخورد با وضعیت T13 در سرویس HA وجود ندارد. به عبارت دیگر، در صورت بروز مشکل در رفع اشکال میزبان از کار افتاده و انتقال ماشین های مجازی آن به دیگر میزبانان در حال کار، پروسه HA دچار مشکل می شود و ماشین های مذکور در دسترس نخواهند بود. این وضعیت ممکن است به دلیل عدم وجود منابع مازاد یا رزرو شده بر روی میزبانان یا خاموش نشدن کامل ماشین های مجازی روی میزبان مختل شده رخ دهد.

باید توجه داشت که هنگام تعریف یک کلاستر HA، سیستم با توجه به میزان منابع آزاد سیستم های میزبان اجازه تعریف کلاستر را می دهد. اما ممکن است منابع مورد تقاضای ماشین های مجازی در حین کار بیشتر از تخمین اولیه باشد. بنابراین در هنگام از کار افتادن یک میزبان، بر خلاف انتظار، منابع کافی برای اضافه شده ماشین های در حال انتقال به میزبانان سالم وجود نداشته باشد. همچنین با توجه به قفل شدن فایل های مربوط به ماشین های مجازی در حال کار، همانطور که گفته شد لازم است در ابتدا ماشین های یک میزبان از کار افتاده یا ایزوله شده توسط وی خاموش شوند تا قفل مربوطه توسط VMFS آزاد شود. این کار در مدل ۴,۷ در گزار T10 انجام می گیرد. اگر در این مرحله به هر دلیل این فرایند خاموش شدن کامل نشود، قفل دیسک آزاد نشده و این ماشین نمی تواند بر روی میزبان دیگری راه اندازی مجدد شود. این اختلال در فرایند خاموش شدن می تواند به اختلال داخلی سیستم عامل ماشین مجازی یا عدم کارکرد صحیح سخت افزارهای مجازی این ماشین مربوط باشد.

در سایر موارد می توان روال منظمی را در انتقال کنترل در این پروسه مشاهده کرد. همچنین لازم به یادآوری است که این مدل تنها روند اجرای سرویس HA در هنگام بروز مشکل را نشان می دهد. بنابراین برای توصیف شیوه افزودن یک میزبان به کلاستر HA و یا تعریف یک کلاستر جدید در حین کار دیتا سنتر، نیاز به طراحی مدل مجزایی می باشد. در قسمت بعدی از این بخش، به تحلیل و ارزیابی مدل طراحی شده در این قسمت خواهیم پرداخت.

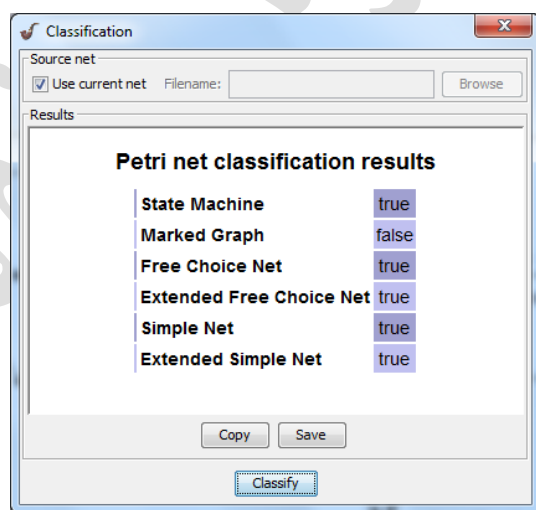
۴,۳,۳. تحلیل و ارزیابی مدل

در این بخش با تحلیل خصوصیات رفتاری مدل میزان انعطاف پذیری و پایداری آن و در نهایت خوش رفتاری سیستم را بررسی خواهیم کرد. به همین منظور خصوصیات Safeness, Liveness و Reversibility را بر روی آن بررسی می نمائیم.

Liveness ۴,۳,۳,۱

برای بررسی خصوصیت Liveness، باید به این نکته توجه داشته باشیم که یک شبکه live از درجه k است در صورتی که تمام گزاره‌های آن شبکه live از درجه k باشند [۲۳]. به این ترتیب روشن ترین راهی که برای تعیین میزان Liveness شبکه های پتری به نظر می رسد، بررسی تک تک گزاره‌های آن از نقطه نظر میزان liveness است. اما به دلیل پیچیده و وقت گیر بودن این شیوه، قضایایی برای اثبات کلیت liveness در شبکه ها، در صورتی که آنها از زیرمجموعه های معینی از شبکه های پتری باشند ارائه شده است. این زیر مجموعه ها که در فصل ۲ معرفی و بررسی شدند، ویژگی های خاصی را به طور مشترک دارا می باشند.

به این ترتیب برای بررسی خصوصیت، در ابتدا سعی می کنیم با بررسی ویژگی های ساختاری شبکه مورد نظر، آن را در یکی از این زیر مجموعه ها طبقه بندی کنیم و سپس با استفاده از قضایای آن زیر مجموعه خصوصیات مذکور از جمله Liveness را راحت تر بر روی آن بررسی نمائیم. با دقت بیشتر خواهیم فهمید که مدل شکل ۴,۷ یک state machine است زیرا هریک از گزاره ها در این شبکه تنها یک ورودی و یک خروجی دارند. عضویت این مدل در دیگر زیرمجموعه های شبکه های پتری در شکل ۴,۸ آمده است. این طبقه بندی توسط نرم افزار PIPE انجام شده است.



شکل ۴,۸. عضویت شبکه ۴,۷ در زیرکلاس های شبکه های پتری

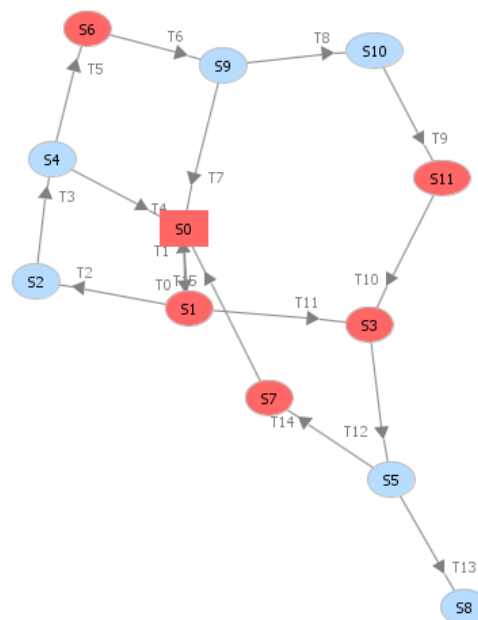
پس با تکیه بر قضیه ۴ از فصل ۲، با توجه به اینکه در وضعیت M_0 یک توکن در شبکه وجود دارد، اگر این شبکه متصل قوی باشد بنابراین live نیز هست.

همچنین با مقایسه قضیه ۱ و بسط آن در همان فصل می توان نتیجه گرفت که شرط لازم برای متصل قوی بودن یک شبکه، عدم وجود موقعیت source و sink و نیز گزار source و sink در شبکه است. در چنین شرایطی شبکه ۴,۷ متصل قوی نمی باشد زیرا P11 در آن یک موقعیت sink محسوب می شود.

در نتیجه براساس قضیه ۴، این شبکه خصوصیت Liveness را ندارد. این امر به این دلیل است که در صورت بروز اختلال در دریافت منابع کافی برای راه اندازی سیستم ثانویه، پروسه HA دچار مشکل خواهد شد. با توجه به این مشکل، مدل متناظر، خاصیت Liveness را از دست می دهد.

Safeness. ۴,۳,۳,۲

در این مرحله با بررسی قضیه ۲ در همان بخش می توانیم خاصیت safeness را در مدل بررسی نمائیم. برای این کار لازم است گراف پوشای مدل را به دست آوریم. با استفاده از نرم افزار PIPE این گراف به صورت شکل ۴,۹ خواهد بود.



شکل ۴,۹. گراف پوشای مدل ۴,۷

با بررسی این گراف در میابیم که برچسب همه گره های آن حاوی مقادیر ۰ و ۱ برای موقعیت های P_0 تا P_{10} هستند. به این معنی که تعداد توکن های موجود در این موقعیت ها، در هیچ وضعیتی بیش از ۱ نبوده است. با این دلیل، مدل همواره 1-bounded یا به عبارت دیگر safe می باشد.

با مقایسه نتیجه حاصل از این دو قضیه می توان فهمید که مدل به دلیل وجود بن بست، دارای خاصیت liveness نیست. به کمک نرم افزار تحلیلگر PIPE می توان مسیری را که مدل طی آن دچار بن بست می شود مشخص کرد. (شکل ۴,۱۰)

به همین منظور و با بررسی گراف شکل ۴,۹، در می یابیم که کلیه وضعیت ها غیر از S8 می توانند کنترل سیستم را به S0 (M_0) بازگردانند. نحوه توزیع توکن در S8 در شکل ۴,۱۱ آمده است.

S8 [Vanishing State]
Marking: {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0}
Edges From: S5 (T13)
Edges To: -

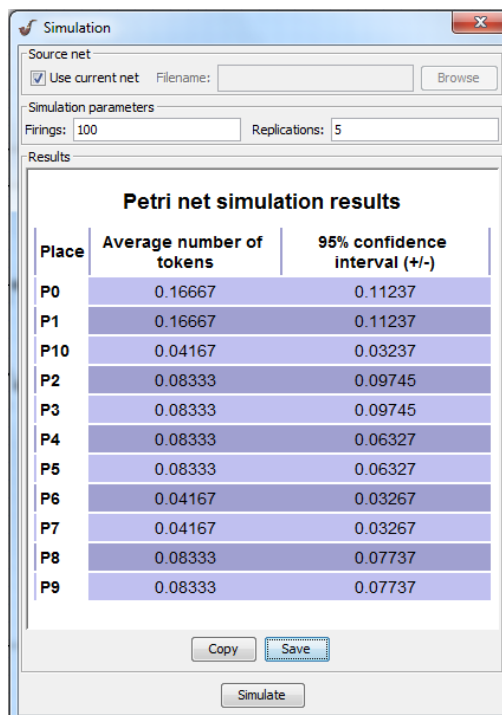
Marking corresponds to {P0, P1, P10, P11, P2, P3, P4, P5, P6, P7, P8, P9}

شکل ۴,۱۱. نحوه توزیع توکن در وضعیت S8

لازم به ذکر است که مفهوم S (State) در نرم افزار مدل سازی PIPE، متناظر با مفهوم M (Marking) یا همان "وضعیت" است که قبلا درمورد آن توضیح داده شده است. با این تفسیر، می توان دریافت که دقیقا زمانی که پس از اجرا شدن T13 توکن در موقعیت P11 قرار می گیرد، سیستم دیگر قابل بازگشت به وضعیت اولیه M_0 نخواهد بود. به این ترتیب می توان دریافت که مدل ۴,۷ دارای خاصیت Reversibility نمی باشد.

■

در مرحله آخر با شبیه سازی اجرای سیستم در چند مرحله متفاوت، نحوه توزیع توکن ها در شبکه را در حین اجرای سیستم بررسی می کنیم. برای این کار در هر مرحله، پنج بار مدل را از نقطه آغاز (P0) اجرا می کنیم و حداکثر ۱۰۰ اجرا بر روی گزاره های مدل انجام می دهیم. نتیجه به صورت شکل ۴,۱۲ خواهد بود.



شکل ۴,۱۲. شبیه سازی شبکه پتری شکل ۴,۷

همانطور که دیده می شود توزیع توکن ها در موقعیت های شبکه به طور یکنواخت انجام شده است که این امر به دلیل یکسان بودن اولویت اجرای گزاره های سیستم است. با این حال تضمینی برای تکرار شدن این نتیجه در اجراهای بعدی وجود ندارد زیرا ممکن است با قرار گیری سیستم در موقعیت P9، گزار T13 اجرا شده و توکن از شبکه حذف شود. همچنین با اجرا شدن T4 و یا T7، همه گزاره های بعد از آنها (T6، T8 و ...) امکان اجرا شدن نخواهند داشت. به همین دلیل بهتر است فرایند شبیه سازی را چندین بار تکرار کنیم تا حالات بیشتری از توزیع توکن ها در شبکه را ببینیم. نتیجه چهار اجرای مختلف از شبیه سازی مدل مذکور در شکل ۴,۱۳ آمده است.

Place	Average number of tokens
P0	0.16667
P1	0.16667
P10	0.04167
P2	0.08333
P3	0.08333
P4	0.08333
P5	0.08333
P6	0.04167
P7	0.04167
P8	0.08333
P9	0.08333

Place	Average number of tokens
P0	0.2
P1	0.2
P10	0
P2	0
P3	0
P4	0
P5	0
P6	0
P7	0
P8	0.2
P9	0.2

Place	Average number of tokens
P0	0.27273
P1	0.27273
P10	0.06061
P2	0.09091
P3	0.09091
P4	0
P5	0
P6	0
P7	0
P8	0.09091
P9	0.09091

Place	Average number of tokens
P0	0.25
P1	0.25
P10	0.08333
P2	0
P3	0
P4	0
P5	0
P6	0
P7	0
P8	0.16667
P9	0.16667

شکل ۴.۱۳. چندین نمونه از شبیه سازی اجرای شبکه پتری

همانطور که دیده می شود، در اجرای دوم، تنها حلقه T_0 ، T_1 اجرا شده اند و در مواقعی نیز با اجرای T_{11} و T_{12} سیستم به وضعیت بن بست رسیده است. در اجرای سوم، با اجرا شدن T_4 امکان قرارگیری توکن در P_4 تا P_7 از بین رفته است. و در نهایت در اجرای چهارم از شبیه سازی، با اجرای T_{11} ، هیچ توکنی در موقعیت های P_2 تا P_7 قرار نمی گیرد. البته باید توجه داشت که هر یک از این مرحله ها با اجرای پنج مرتبه این مدل به دست آمده اند.

در صورت تغییر اولویت اجرای گزارها در سیستم می توان میزان حضور توکن را در موقعیت ها تغییر داد. اما این تغییر در اولویت باید بر مبنای یک الگوی قابل تعریف باشد.

۴.۴. بررسی نحوه کار سرویس Fault Tolerance

سرویس Fault Tolerance (FT) به منظور ایجاد اطمینان از همیشه در دسترس بودن ماشین های مجازی حیاتی که بر روی میزبانان در حال کار هستند طراحی شده است. این میزبانان در حال سرویس دادن به ماشین های خود از طریق ESX هستند. سرویس FT سطح بالاتری از دسترس پذیری را نسبت به سرویس HA فراهم می کند به این معنی که مدت زمان خرابی سیستم در هنگام استفاده از این شیوه بسیار کاهش می یابد.

در ادامه این بخش به بررسی ساختار فنی FT و نحوه سرویس دهی آن می پردازیم [۶۱].

۴.۴.۱. تشریح ساختار سرویس Fault Tolerance

سرویس FT به کمک ایجاد و نگهداری یک ماشین مجازی ثانویه، امکان دسترسی پذیری دائمی را برای ماشین های مجازی مورد نظر به وجود می آورد. این کپی ثانویه از ماشین مجازی اصلی کاملاً همسان با ماشین اولیه بوده و مانند آن همواره در حال اجرا و در دسترس است. ماشین ثانویه همیشه آماده جایگزین شدن به جای ماشین اولیه در صورت بروز هرگونه اختلال برای نمونه اصلی است.

مدیر سیستم می تواند این امکان را برای ماشین های مجازی حساس و حیاتی که همیشه باید در حال سرویس دهی باشند راه اندازی نماید. نسخه کپی شده ماشین اصلی که ماشین ثانویه خوانده می شود توسط سرویس FT روی میزبانی متفاوت با میزبان ماشین اصلی ایجاد شده و به موازات ماشین اصلی کار می کند. سرویس VMware vLockstep تمام ورودی ها و اتفاقات در حال وقوع روی ماشین اصلی را ضبط کرده و برای ماشین ثانویه می فرستد. با این اطلاعات ماشین ثانویه قدم به قدم مطابق ماشین اولیه دستورات را اجرا می کند.

ماشین های اصلی و ثانویه به طور مداوم سیگنال های ضربانی را با یکدیگر تبادل می نمایند. تبادل سیگنال به این جفت ماشین مجازی امکان می دهد که وضعیت یکدیگر را مانیتور کرده و از اجرای دقیق سرویس FT مطمئن شوند. در صورتی که ماشین مجازی اصلی به هر دلیل از کار بیفتد یا از مدار خارج شود یک فرایند رفع اشکال به صورت پنهان از دید بقیه اجزای سیستم و نیز کاربران

انجام خواهد شد.^۱ به این معنی که به صورت آنی ماشین ثانویه جای ماشین اولیه را گرفته و به جای آن وظایف محوله را انجام و به درخواست ها پاسخ می دهد. همانطور که پیشتر گفته شد، کاربران هیچ تاخیر یا وقفه ای را در دریافت سرویس از ماشین مربوطه حس نخواهند کرد.

همانطور که می توان حدس زد، ماشین مجازی انتخاب شده برای اعمال سرویس FT و نیز کپی ثانویه آن نمی توانند بر روی یک میزبان واحد اجرا شوند. زیرا در صورت بروز اختلال در میزبان مذکور، هر دو عضو این جفت FT از کار خواهند افتاد. به همین دلیل سرویس FT قوانین جلوگیری از اجتماع همبستگان^۲ را در مورد جفت های FT اجرا می کند تا از اجرای ماشین های مجازی عضو یک جفت روی میزبانان جداگانه مطمئن شود.

در این میان احتمال بروز یک خطر سرور مذکور را تهدید می کند و آن احتمال بروز پدیده "دو مغزی"^۳ در مورد سرور است. با این توضیح که اگر پس از انتقال کنترل اجرا و سرویس دهی از ماشین اصلی به ماشین ثانویه، ماشین اصلی رفع اشکال شده و دوباره به مدار برگردد و شروع به سرویس دهی نماید، در این حالت ما دو کپی از یک ماشین را خواهیم داشت که هر دو در حال پاسخگویی به یک نوع درخواست هستند. برای جلوگیری از این حالت از قفل کردن فایل مربوط به ماشین اصلی بر روی منبع ذخیره سازی مشترک استفاده می شود تا رفع ایراد ماشین را با اجرای ماشین فعلی هماهنگ کرده و تنها اجازه اجرای یک ماشین را به عنوان ماشین اصلی در یک جفت FT بدهد. بنابراین پس از انتقال کنترل اجرا از ماشین اصلی از کار افتاده به ماشین ثانویه در حال کار، این ماشین به عنوان عضو اصلی ثبت شده و یک ماشین مجازی دیگر به عنوان ماشین ثانویه جدید کپی می شود.

موارد کاربرد سرویس Fault Tolerance

^۱ Transparent failover

^۲ Anti-affinity

^۳ Split-brain

استفاده از سرویس FT در چند حالت می تواند برای ایجاد پایداری و دسترس پذیری ماشین های مجازی پر اهمیت و مفید باشد.

با توجه به اینکه سرویس FT سطح بالاتری از پیوستگی اجرایی^۱ را نسبت به سرویس HA در اختیار سیستم قرار می دهد، می تواند برای اطمینان از فعال بودن دائمی سرویس های بسیار ضروری سازمان مورد استفاده قرار گیرد. وقتی که از ماشین ثانویه خواسته می شود که جای ماشین اصلی را بگیرد، این ماشین به سرعت در وضعیت ماشین اصلی قرار گرفته و تمام وظایف وی را انجام می دهد. برنامه های اجرایی و سرویس ها در این لحظه بر روی ماشین ثانویه در حال اجرا هستند و داده های لازم بر روی حافظه کاری این ماشین وجود دارد زیرا در واقع حافظه ماشین ثانویه آینه ای از حافظه ماشین اولیه بوده است.

این شیوه، تفاوت اساسی را در مقایسه با روش سرویس HA در رفع اشکال ماشین ها ایجاد می کند. زیرا در مکانیزم HA نیاز به زمان بیشتری برای راه اندازی دوباره ماشین ها بر روی یک میزبان جدید وجود دارد که بخش زیادی از این زمان نیز به مدت زمان لازم برای بوت شدن سیستم عامل ماشین بستگی دارد. همچنین در صورت ایزوله شدن یک میزبان، باید زمان لازم برای خاموش شدن ماشین های مجازی بر روی میزبان ایزوله شده برای آزاد شدن قفل مربوط به فایل ماشین مذکور را نیز به زمان تاخیر اضافه نمود. این در حالی است که مدت زمان تاخیر برای سرویس FT در صورت بروز خرابی، صفر خواهد بود.

در مقابل باید توجه داشت که این سطح از دسترس پذیری و امنیت اجرایی سرویس ها، مستلزم نگهداری و اجرا کردن دو نسخه از یک ماشین مجازی است که هر یک منابع کافی را برای اجرا طلب می کنند. بنابراین تعریف جفت های FT باید طبق یک برنامه مدون و با توجه به منابع موجود سخت افزاری صورت پذیرد.

سرویس FT در مورد سناریوهای زیر می تواند بسیار سودمند باشد.

¹ Business continuity

- برنامه های کاربردی که لازم است همیشه در دسترس باشند. به خصوص آنهایی که اتصالات طولانی مدت با سیستم کاربران برقرار می کنند و اختلال در سرویس می تواند باعث ایجاد قطعی کامل سرویس دهی به کاربر شود.
- نرم افزارهای کاربردی که روش دیگری برای ایجاد کلاستر در مورد آنها وجود ندارد.
- حالت هایی که امکان high availability باید از طریق شیوه های اختصاصی خود نرم افزار انجام شود و پیکر بندی کردن با این شیوه ها بسیار پیچیده و هزینه بر است.

ارائه سرویس Fault Tolerance بر حسب مورد^۱

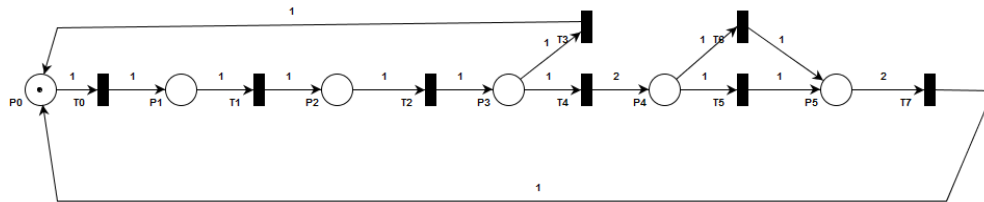
یکی دیگر از موارد مورد نیاز در استفاده از سرویس FT مواردی است که یک ماشین مجازی در بازه های زمانی خاصی در حال ارائه سرویس ضروری می باشد ولی در مواقع دیگر سرویس این ماشین چندان حیاتی نیست. در چنین حالاتی احتمالاً ماشین مذکور به طور عادی توسط یک کلاستر HA محافظت می شود ولی در این مواقع حساس نیاز به مراقبت و دسترس پذیری بیشتری دارد.

به عنوان مثال وقتی مدیر سیستم تصمیم دارد یک اخبار تلویزیونی را در ساعت پربیننده پخش کند، سرویس ماشین مربوطه باید در طی چند دقیقه پخش اخبار به شدن محافظت شده و از کار نیافتد. اما در ساعات دیگر این سرویس مورد نیاز نیست. به کمک یک امکان در سرویس FT، مدیر می تواند در زمان پخش اخبار ماشین مجازی مربوط به این سرور را با سطح بالاتری از دسترس پذیری حفاظت نماید و پس از پایان برنامه سرویس FT را بر روی این ماشین غیر فعال کرده و آن را به وضعیت عادی مراقبتی بازگرداند.

۴.۴.۲. ارائه مدل فرمال از نحوه کار سرویس Fault Tolerance

^۱ On-Demand Fault Tolerance

در این بخش به ارائه مدلی فرمال برای تحلیل رفتار سرویس Fault Tolerant خواهیم پرداخت. این مدل شبکه پتری در شکل شماره ۴،۱۴ دیده می شود.



شکل ۴،۱۴. مدل پتری نحوه کار سرویس Fault Tolerance

در ادامه به شرح گزارها و مراحل تغییر حالت در سیستم فوق می پردازیم.

T0: سرویس vLockstep ورودی ها و وقایع ماشین مجازی اصلی را ثبت می کند

T1: سرویس vLockstep وضعیت ثبت شده را برای سرویس مشابه در ماشین مجازی ثانویه

ارسال می کند

T2: ماشین ثانویه بر اساس صورت وضعیت رسیده، فرمان های مشابه را عینا اجرا می کند

T3: بر اساس سیگنال ضربانی تبادل شده، هر دو ماشین اصلی و ثانویه سالم و در حال کار

هستند.

T4: ماشین اصلی دچار اختلال شده و سیگنال ضربانی را برای ماشین ثانویه ارسال نمی کند.

T5: ماشین ثانویه جایگزین ماشین اصلی شده و به جای آن شروع به سرویس دهی می نماید

T6: فایل مربوط به ماشین اولیه توسط VMFS قفل می شود تا از پدیده دو مغزی در ماشین

مجازی جلوگیری شود.

T7: ماشین ثانویه که در حال سرویس دهی می باشد به عنوان ماشین اصلی معین می شود و

یک ماشین ثانویه جدید به صورت کپی از آن ایجاد شده و شروع به کار می کند.

باید توجه داشت که مدل فوق، در واقع نحوه انتقال کنترل در حین اجرای نرمال سرویس FT

می باشد. بنابراین پروسه های مربوط به پیکربندی سرویس نیاز به طراحی مدل های جداگانه دارد.

بنابراین مدل های دیگری برای ایجاد جفت FT در مرحله شروع کار دیتا سنتر، ایجاد جفت FT جدید در حین کار دیتا سنتر و نیز راه اندازی سرویس FT بر حسب مورد، باید طراحی شود. همچنین در این مدل، در موقعیت P4 سیستم با انجام همزمان دو مسئولیت به طور موازی رو بروست. این دو وظیفه شامل انتقال کنترل سرویس دهی از ماشین اصلی به ماشین ثانویه و نیز قفل کردن ماشین اولیه برای جلوگیری از سروس دهی مجدد، همزمان با ماشین ثانویه می باشد. برای مدل سازی این همزمانی در اجرای پروسه لازم است وزن لبه ارتباطی از T4 به P4 برابر ۲ باشد به این معنی که T4 پس از fire شدن دو توکن را در P4 قرار دهد تا به این ترتیب هر دو گزار T5 و T6 فعال شده و امکان fire شدن داشته باشند. همچنین پس از قرار گرفتن سیستم در موقعیت P5 و اجرای دو وظیفه فوق، برای جلوگیری از بروز سرریز در این موقعیت، لازم است با fire شدن گزار T7 هر دو توکن موجود در این موقعیت مصرف شوند. بنابراین وزن لبه ارتباطی بین P5 و T7 برابر ۲ می باشد.

۴,۴,۳. تحلیل و ارزیابی مدل

در این بخش با تحلیل خصوصیات رفتاری مدل، میزان انعطاف پذیری و پایداری آن و در نهایت خوش رفتاری سیستم را بررسی خواهیم کرد. به همین منظور خصوصیات Safeness, Liveness و Reversibility را بر روی آن بررسی می نمائیم. در بررسی خصوصیات این شبکه باید توجه داشت که شبکه از نوع عادی نمی باشد زیرا در لبه های T4-P4 و نیز P5-T7، وزن لبه ها بیش از ۱ می باشد. بنابراین این شبکه در زیر کلاس های شبکه های پتری، مطرح در فصل ۲، قرار نمی گیرد.

۴,۴,۳,۱. Liveness

با توجه به عدم امکان استفاده از قضایای مربوط به زیرکلاس های شبکه های پتری، تنها راه ممکن برای تشخیص سطح liveness این شبکه بررسی سطح liveness تک تک گزارهای سیستم است.

T0, T1, T2: L4-live, برای اجرا شدن هر دنباله ای از وضعیت ها (M_1 تا M_n) این گزارها باید اجرا شوند.

T3, T4: L3-live, در M_3 (S3 در گراف پوشای شکل ۴,۱۶) یکی از این دو گزار قطعا اجرا می شود. بنابراین این گزارها به دفعات نامحدود می توانند اجرا شوند.

T5, T6: L3-live, در صورت قرارگیری سیستم در M_4 , حتما هر دو گزار اجرا خواهند شد. بنابراین آنها می توانند به دفعات دلخواه اجرا شوند. البته زمان اجرای آنها و ترتیب تمام شدن پروسه داخلی آنها لزوما برابر نیست.

T7: L3-live, در صورت اجرا شدن T5 و T6 و قرار گیری سیستم در وضعیت M_6 , این گزار قطعا اجرا می شود. اما به دلیل اینکه ممکن است سیستم با اجرای دنباله M_0, M_1, M_2, M_3 و M_0 اصلا به سراغ اجرای گزارهای T4 به بعد نرود، این گزار نیز فقط به دفعات دلخواه (و نه همیشه) می تواند اجرا شود.

وضعیت قرارگیری توکن ها در M_3 و M_4 در شکل ۴,۱۵ دیده می شوند.

S3 [Vanishing State] Marking: {0, 0, 0, 1, 0, 0} Edges From: S2 (T2) Edges To: S4 (T4); S0 (T3)	S4 [Vanishing State] Marking: {0, 0, 0, 0, 2, 0} Edges From: S3 (T4) Edges To: S5 (T6)
--	---

Marking corresponds to {P0, P1, P2, P3, P4, P5}

شکل ۴,۱۵. نحوه توزیع توکن ها در M_3 و M_4

با توجه به وضعیت liveness گزارهای این شبکه می توان نتیجه گرفت که شبکه در حالت L3-live قرار دارد.

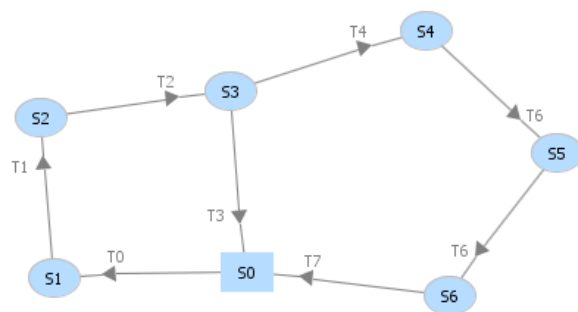
Safeness .۴,۴,۳,۲

با دقت در نحوه انتقال کنترل در مدل می بینیم که با اجرا شدن $T4$ و وزن ۲ برای لبه آن، تعداد ۲ توکن در $P4$ قرار می گیرد. این حالت به $T5$ و $T6$ اجازه اجرای همزمان را می دهد و پس از آن دوباره تعداد ۲ توکن در $P5$ جای می گیرد که با اجرا شدن $T7$ هر دو مصرف می شود. پس در بخشی از شبکه بیش از یک توکن وجود دارد. این حالت، خصوصیت 1-bounded بودن شبکه را از بین برده و شبکه دیگر safe نخواهد بود. البته با توجه به اینکه تعداد توکن های موجود در موقعیت های شبکه هیچ گاه بیشتر از ۲ نخواهد بود، شبکه همچنان bounded باقی می ماند.

Reversibility .۴,۴,۳,۳

برای بررسی این خصوصیت لازم است به تعریف Reversibility توجه کنیم. با یاد آوری تعریف این مفهوم و نیز خاصیت Reachability از فصل ۲ داریم: شبکه پتری (N, M_0) Reversible خوانده می شود اگر برای هر وضعیت M ، در دنباله قابل اجرا از M_0 $(R(M_0))$ ، M_0 از طریق M ، reachable باشد.

با این تعریف، در شبکه های کوچک می توان با بررسی کلیه وضعیت های محتمل در شبکه، reachable بودن M_0 را از طریق آنها بررسی نمود. در صورتی که M_0 از طریق تک تک این وضعیت ها (M_1 تا M_n) قابل دسترسی باشد این شبکه Reversible است. با توجه به اینکه همه وضعیت های قابل وقوع در یک شبکه به عنوان یک گره در گراف پوشای مدل منعکس می شوند، می توان با بررسی یک به یک گره های گراف این حالات را بررسی نمود. شکل ۴,۱۶ گراف پوشای شبکه پتری شکل ۴,۱۴ را نشان می دهد.



شکل ۴،۱۶. گراف پوشای مدل پتری شکل ۴،۱۴

همانطور که در این گراف جهت دار دیده می شود، سیستم همیشه از S_0 شروع شده و با ادامه، به حالات S_1 ، S_2 و S_3 می رود. پس از آن یا به S_0 بر می گردد و یا با گذراندن حالات S_4 ، S_5 و S_6 دوباره به S_0 می رود. بنابراین تمام موقعیت های ممکن در این شبکه (M_1 تا M_6) لزوماً به M_0 برمی گردند. پس این شبکه Reversible است.

۴،۵. بررسی نحوه کار سرویس VMotion

سرویس VMware VMotion به مدیر دیتا سنتر اجازه می دهد که ماشین های مجازی در حال کار را از یک میزبان به میزبان دیگر منتقل کند. طبیعتاً در این میان دو میزبان باید با یکدیگر سازگار باشند. همچنین این انتقال بدون قطعی سرویس و از کار افتادن ماشین مجازی انجام خواهد شد. در ادامه به بررسی دقیقتر مکانیزم این سرویس خواهیم پرداخت [۶۳] و [۷۰].

۴،۵،۱. تشریح ساختار سرویس VMotion

¹ Compatible

این سرویس در واقع بخشی از نرم افزار مدیریتی VirtualCenter است که به مدیر اجازه می دهد ماشین های مجازی در حال کار را از یک میزبان فیزیکی به میزبان فیزیکی دیگر منتقل کند. این کار ممکن است به دلایل زیادی از طرف مدیر صورت پذیرد. از آن جمله می توان به موارد زیر اشاره کرد:

- انجام مراحل تست و نگهداری سخت افزار سیستم میزبان بدون نیاز به قطع سرویس دهی ماشین های مجازی در حال کار بر روی آن
 - اقدام پیشگیرانه برای انتقال ماشین ها از روی یک میزبان ضعیف یا در حال از کار افتادن یا میزبانی که منابع پردازشی کافی برای اجرا کردن ماشین های در حال کار بر روی خود را ندارد.
 - انتقال ماشین ها از میزبانی به میزبان دیگر برای برقراری تعادل و تخصیص بهینه منابع میزبانان به ماشین ها
- برای تشریح بهتر موضوع، در ابتدا لازم است چند اصطلاح را تعریف نمائیم.

انتقال (مهاجرت)^۱ یک ماشین مجازی

به فرایند انتقال کامل یک ماشین مجازی از یک سیستم فیزیکی (میزبان) به سیستم فیزیکی دیگر گفته می شود. مجازی سازی کامل و مستقل از سخت افزار همه بخش های ماشین از جمله پردازنده، BIOS، دیسک، رابط شبکه و حافظه، این امکان را به سیستم می دهد که ماشین مجازی را با کلیه وضعیت اجرایی و سخت افزاری جاری به صورت مجموعه ای از فایل ها ثبت و نگهداری نماید. بنابراین انتقال یک ماشین مجازی از یک میزبان به دیگری، در واقع، انتقال مقداری داده بین این دو میزبان خواهد بود.

بر اساس وضعیت این ماشین مجازی در حین انتقال، به طور کلی می توان مهاجرت را به دو نوع انتقال سرد و انتقال گرم تقسیم کرد.

¹ Migration

انتقال سرد یک ماشین مجازی

در این روش در ابتدا ماشین مجازی روی سیستم میزبان مبدا خاموش شده و سپس به میزبان مقصد منتقل می شود. وقتی انتقال کامل شد، در مقصد دوباره این ماشین مجازی روشن شده و شروع به کار می کند.

انتقال گرم ماشین مجازی

در این روش، ماشین مجازی در حالی که روشن است انتقال می یابد. ماشین مجازی و برنامه های کاربردی در حال اجرا بر روی آن به طور پیوسته بر روی ماشین مبدا در دسترس خواهند بود تا زمانی که این ماشین به مقصد منتقل شود. بنابراین کاربران پس از انتقال هیچ تغییری را حس نخواهند کرد. بر اساس سطح دسترس پذیری ماشین در طول زمان انتقال، این تغییر به دو صورت می تواند انجام گیرد:

- انتقال به صورت معلق کردن و سپس از سر گرفتن کار ماشین؛¹ که در این حالت همانطور که از عنوان آن مشخص است کار ماشین به صورت تعلیق در می آید تا بتوان آن را انتقال داد. پس از کامل شدن انتقال به میزبان مقصد، کار ماشین از سر گرفته می شود.
- انتقال زنده؛² در این شیوه بدون اینکه اختلالی در کار ماشین و سرویس های در حال ارائه توسط آن به وجود آید، از مبدا به مقصد منتقل می شود. و به این ترتیب مدت زمان قطعی سرویس به صفر می رسد. این کار به کمک سرویس VMotion انجام می گیرد.

شیوه کار سرویس VMware VMotion

¹ Suspend/resume migration

² Live migration

انتقال یک ماشین مجازی در حال کار از یک میزبان به میزبان دیگر بدون ایجاد قطعی در روند عادی کار آن شامل سه مرحله اصلی است [۷۰]:

در مرحله اول وضعیت جاری ماشین مجازی به صورت مجموعه ای از فایل ها ثبت شده و بر روی سیستم ذخیره سازی مشترک ذخیره می شود. این منبع ذخیره سازی می تواند دیسک های SCSI مبتنی بر فیبر نوری، دیسک های SCSI مبتنی بر شبکه (iSCSI)، شبکه های تخصصی ذخیره سازی داده ها (SAN) و یا تجهیزات ذخیره سازی تحت شبکه (NAS) باشد.

همانطور که قبلا در همین بخش گفته شد، این وضعیت جاری شامل وضعیت سخت افزارهای مجازی ماشین مذکور به علاوه وضعیت دقیق حافظه ماشین می باشد که نشان دهنده وضعیت پروسه های در حال اجرا در این لحظه بر روی ماشین است.

سرویس VMware vStorage از زیر مجموعه سرویس های VMFS اجازه دسترسی چندین نسخه ESX را به طور همزمان به فایل های یک ماشین مجازی در شرایط خاص و کنترل شده می دهد.

در مرحله دوم، بخش فعال حافظه و وضعیت دقیق اجرایی ماشین مجازی از طریق خطوط شبکه پرسرعت منتقل می شود. این عمل باعث می شود ماشین مذکور به سرعت از حالت اجرایی روی ESX مبدا به ESX مقصد سوئیچ کند. برای اینکه این انتقال کاملا پنهان از دید کاربر باشد، سرویس VMotion تغییراتی را که در حین انتقال وضعیت ماشین از مبدا به مقصد در حافظه ماشین در حال کار در مبدا رخ می دهد را به صورت bitmap ذخیره می کند. وقتی تمام حافظه و وضعیت جاری سیستم به میزبان مقصد کپی شد، VMotion ماشین مبدا را به حال تعلیق درآورده، bitmap تهیه شده از تغییرات نهایی حافظه را روی ماشین مقصد اعمال و ماشین ایجاد شده بر روی میزبان مقصد را راه اندازی می نماید. بر اساس تست های گرفته شده، مجموع زمان لازم برای این مرحله در یک شبکه Gigabit Ethernet کمتر از ۲ ثانیه است.

در مرحله سوم، رابط های مجازی شبکه که قبلا توسط ماشین مجازی استفاده می شدند توسط ESX جدید دوباره تعریف می شوند. به این ترتیب ESX مقصد اطمینان حاصل می کند که پس

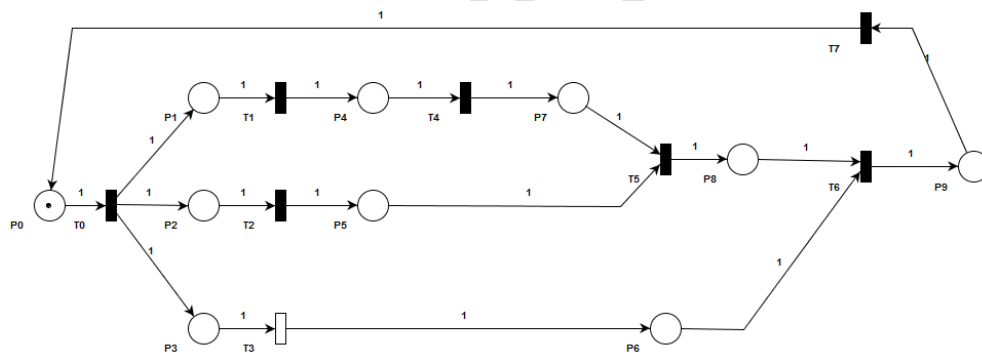
¹ vNIC

از انتقال ماشین، باز هم اتصالات شبکه آن برقرار و آماده کار است. VMotion انتقال آدرس فیزیکی MAC را به عنوان بخشی از پروسه انتقال مدیریت می کند. وقتی ماشین بر روی میزبان مقصد فعال شد، VMotion آدرس روتر ماشین را ping می کند تا از شناخته شدن ماشین توسط روتر در محل جدید و با این MAC مطمئن شود.

به دلیل اینکه انتقال یک ماشین مجازی به کمک سرویس VMotion با انتقال کامل وضعیت اجرایی ماشین، مشخصات کارت های شبکه و اتصالات فعال شبکه انجام می شود در نتیجه کاربر هیچ اختلال و قطعی را در سرویس دهی ماشین مذکور حس نخواهد کرد.

۴,۵,۲. ارائه مدل فرمال از نحوه کار سرویس VMotion

در این بخش به ارائه مدلی فرمال برای تحلیل رفتار سرویس VMotion خواهیم پرداخت. این مدل شبکه پتری در شکل ۴,۱۷ دیده می شود.



شکل ۴,۱۷ مدل پتری نحوه کار سرویس VMotion

همانند بخش های قبل، این مدل نیز نشان دهنده جزئیات بیشتری از طرز کار سیستم است که در مدل کلی شکل ۴,۴ نشان داده شده است. این شکل در واقع نمود دقیقتری از گزار T41 در شکل

¹ Default Gateway

۴,۴ است. بنابراین موقعیت های P0 و P6 در این شکل به ترتیب متناظر با P30 و P31 در شکل ۴,۴ هستند.

در ادامه به شرح مراحل تغییر وضعیت مدل در سیستم طرح شده می پردازیم.
T0: آخرین تغییرات ماشین مجازی در فایل متناظر آن با پسوند vmdk ذخیره می شود.
T1: ساختن یک تصویر دقیق از وضعیت حافظه ماشین مبدا
T2: راه اندازی ماشین مذکور بر روی میزبان مقصد به کمک فایل ماشین مجازی با پسوند vmdk.
T3: شروع فرایند دنبال سازی دقیق تغییرات در حال رخداد بر روی حافظه در طول انتقال ماشین مجازی^۱

T4: انتقال تصویر حافظه ایجاد شده در گزار T1 به میزبان مقصد
T5: به حال تعلیق درآوردن ماشین مجازی بر روی میزبان مبدا
T6: کپی کردن تصویر حافظه به دست آمده در گزار T3 به ماشین مجازی در مقصد و استارت کردن ماشین
T7: ارائه feedback به VMkernel

باید توجه داشت که بر خلاف سرویس هایی چون HA و Fault Tolerance، این سرویس خودکار نمی باشد. به عبارت دیگر سیگنال شروع این سرویس و انتقال یک ماشین مجازی از یک میزبان به میزبان دیگر نه بر مبنای شرایط خاص محیطی مانند مختل شدن یک میزبان، بلکه بر اساس دستور مدیر سیستم یا یک سرویس دیگر می باشد. بنابراین در مدل لایه یک سیستم قبل از شرع این فرایند یک گزار شامل درخواست سرویس فوق توسط کاربر وجود دارد که در سرویس های دیگر به چشم نمی خورد.

در قسمت بعدی از این بخش به تحلیل مدل ارائه شده خواهیم پرداخت.

۴,۵,۳. تحلیل و ارزیابی مدل

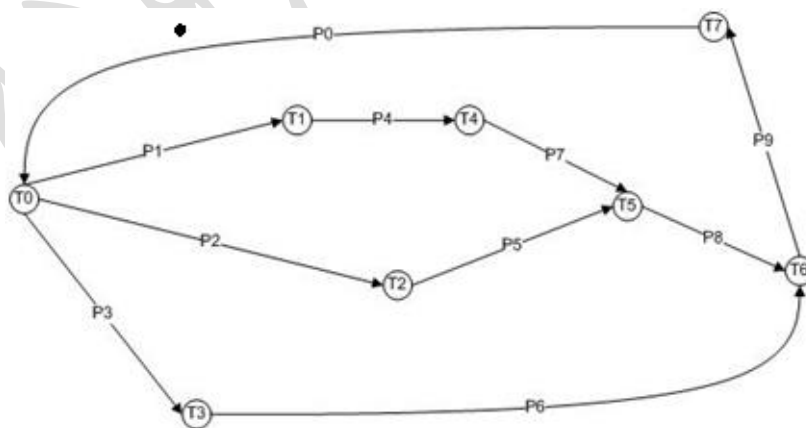
¹ On-going memory transaction bitmap capture

در این بخش با تحلیل خصوصیات رفتاری مدل میزان انعطاف پذیری و پایداری آن و در نهایت خوش رفتاری سیستم را بررسی خواهیم کرد. به همین منظور خصوصیات *Safeness*, *Liveness* و *Reversibility* را بر روی آن بررسی می نمائیم.

برای بررسی خصوصیات این شبکه پتری بهتر است آن را در یکی از زیر مجموعه های شبکه های پتری طبقه بندی کنیم. به این ترتیب بررسی و اثبات خواص آن ساده تر خواهد شد. با بررسی دقیق این مدل در می یابیم که تمام موقعیت ها در شبکه دقیقا دارای یک لبه ورودی و یک لبه خروجی هستند. این خاصیت، شبکه را در گروه *marked graph* جای می دهد.

Liveness. ۴,۵,۳,۱

برای اثبات *liveness* مدل شکل ۴,۱۷ می توانیم از قضیه ۷ در فصل ۲ استفاده کنیم. بر اساس این قضیه، اگر نشان دهیم که در گراف نشانه دار (G, M_0) که از روی (N, M_0) ترسیم می شود، در وضعیت M_0 در هر مدار جهت دار حداقل یک توکن وجود دارد، این شبکه *live* خواهد بود. به این منظور گراف نشانه دار مدل مذکور در شکل ۴,۱۸ آمده است. با توجه به اینکه هیچ گزار یا موقعیتی که خاصیت *source* و یا *sink* داشته باشد وجود ندارد، می توان نتیجه گرفت که همه موقعیت ها و گزارها در شبکه عضو یک مدار جهت دار هستند.



شکل ۴,۱۸. گراف نشانه دار (G, M_0) مربوط به مدل ۴,۱۷

همچنین با توجه به توضیح مدار مستقیم، مدارهای گراف G را می توان به شرح زیر تشخیص داد:

- $P0, P1, P4, P7, P8, P9$
- $P0, P2, P5, P8, P9$
- $P0, P3, P6, P9$

به این ترتیب با حضور $P0$ در تمام آنها و نیز با وجود یک توکن در $P0$ در وضعیت M_0 ، می توان نتیجه گرفت که در تمام مدارهای مستقیم گراف G در وضعیت ابتدا (M_0) یک توکن وجود دارد. در چنین شرایطی، بر اساس قضیه ۶ می توان نتیجه گرفت که در تمام موقعیت های مدل بدون توجه به ترتیب اجرای گزارها، یک توکن وجود خواهد داشت. این امر به طور تلویحی به $live$ بودن گزارها اشاره می کند. همچنین قضیه ۷ صریحا خصوصیت liveness شبکه را که نتیجه $live$ بودن گزارهای آن است نتیجه می گیرد.

Safeness. ۴,۵,۳,۲

برای اثبات این خصوصیت بر روی مدل این بخش، رجوع به قضیه ۶ کافی است. زیرا طبق این قضیه تعداد توکن ها در هر مدار جهت دار در یک $marked\ graph$ مستقل از ترتیب اجرا شدن گزارهای آن مدار و برابر با تعداد توکن ها در وضعیت اولیه M_0 است. همچنین با توجه به اینکه شبکه $live$ است، می توان مطمئن بود که همه گزارهای آن در وضعیت های محتمل اجرا می شوند. در نهایت چون در M_0 تنها یک توکن در همه مدارهای جهت دار گراف وجود دارد، در هر دنباله از اجرای گزارها این مقدار تغییر نخواهد کرد.

با این حال برای اثبات صریح وجود صفت Safeness در شبکه مذکور از قضیه ۸ استفاده می کنیم. طبق این قضیه یک $marked\ graph$ که دارای خاصیت liveness می باشد فقط در صورتی

¹ Possible markings

safe است که هر موقعیتی در آن متعلق به یک مدار جهت دار مانند C باشد که تعداد توکن ها در C در وضعیت M_0 برابر با ۱ است ($M_0(C)=1$).

در گام اول، این شبکه live است. در گام دوم باید به حلقه های جهت دار مطرح شده در بخش قبل توجه کرد. می بینیم که همه موقعیت های شبکه در حداقل یکی از این حلقه ها عضویت دارند. همچنین در وضعیت M_0 ، تنها موقعیت دارای توکن در هر سه حلقه، موقعیت P0 است که دارای ۱ توکن است. بنابراین داریم:

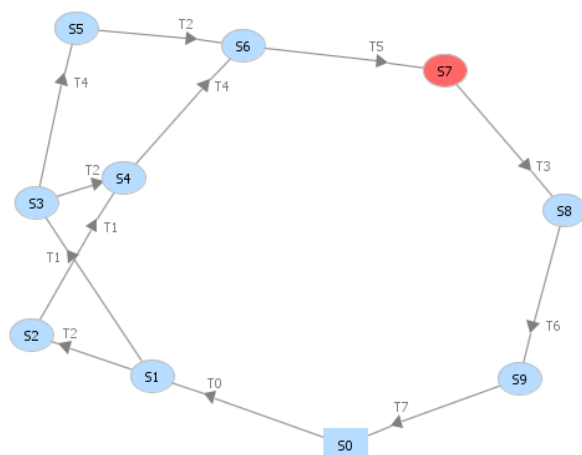
$$M_0(C1) = M_0(C2) = M_0(C3) = 1$$

بنابراین شبکه پتری مذکور safe است.

Reversibility. ۴,۵,۳,۳

برای بررسی این خصوصیت، همانند بخش های قبل، لازم است به تعریف Reversibility توجه کنیم. با یاد آوری تعریف این مفهوم و نیز خاصیت Reachability از فصل ۲ داریم: شبکه پتری (N, M_0) ، Reversible خوانده می شود اگر برای هر وضعیت M، در دنباله قابل اجرا از M_0 $(R(M_0), M_0)$ از طریق M reachable باشد.

با این تعریف، در شبکه های کوچک می توان با بررسی کلیه وضعیت های محتمل در شبکه، reachable بودن M_0 را از طریق آنها بررسی نمود. در صورتی که M_0 از طریق تک تک این وضعیت ها (M_1 تا M_n) قابل دسترسی باشد این شبکه Reversible است. با توجه به اینکه همه وضعیت های قابل وقوع در یک شبکه به عنوان یک گره در گراف پوشای مدل منعکس می شوند، می توان با بررسی یک به یک گره های گراف این حالات را بررسی نمود. شکل ۴,۱۹ گراف پوشای شبکه پتری شکل ۴,۱۷ را نشان می دهد.



شکل ۴،۱۹. گراف پوشای مدل ۴،۱۷

همانطور که در شکل دیده می شود، از هریک از وضعیت های S1 تا S9 با اجرای یک دنباله از گزارها می توان به وضعیت اولیه S0 (M_0) رسید. بنابراین می توان نتیجه گرفت که شبکه مذکور دارای خاصیت Reversibility می باشد.

■
با توجه به خواص بررسی شده در مورد شبکه پتری مربوط به سرویس VMotion، و به این دلیل که این شبکه دارای خواص liveness، safeness و reversibility می باشد، می توان نتیجه گرفت که مدل مذکور خوش رفتار^۱ است. این موضوع تا حد زیادی ما را در مورد قابل کنترل و قابل پیش بینی بودن رفتار سیستم مطمئن می سازد.

¹ Well-behaved

۴,۶. بررسی ساختار داخلی ESX hypervisor

در این بخش به تحلیل ساختار و مدل سازی نرم افزار ESX که نوعی hypervisor است خواهیم پرداخت. این نرم افزار در واقع نوعی سیستم عامل است که ضمن مدیریت منابع سیستم میزبان، سرویس های لازم را نیز به ماشین های مجازی ارائه می دهد [۳] و [۵۹].

۴,۶,۱. تشریح ساختار ESX

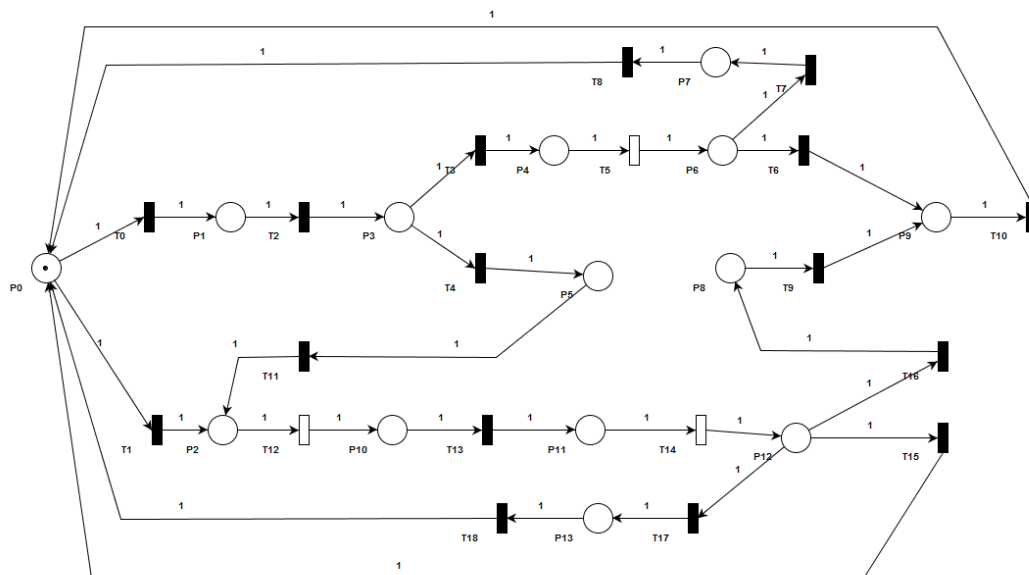
در این حوزه با توجه به ماهیت سیستم عاملی این نرم افزار، انتظار می رود که ESX بتواند مدیریت منابع سخت افزاری، نرم افزاری ورودی و خروجی و همچنین مدیریت پروسه ها را انجام دهد. همچنین باید توجه داشت که هدف از اجرای این سیستم عامل بر روی ماشین میزبان، ایجاد تصویری انتزاعی از منابع سیستم از جمله پردازنده و حافظه است تا ماشین های مجازی بتوانند با این تصویر انتزاعی کار کنند. بنابراین لازم است ESX کمترین بار پردازشی ممکن را داشته باشد. به عبارت دیگر باید به صورت لایه بسیار نازکی بر روی سخت افزار کار کند.

ESX به طور مستقیم بر روی سخت افزار قرار گرفته و به عنوان یک لایه مجازی سازی قوی بین سخت افزار و سیستم عامل های میهمان ایفای نقش می کند. در واقع ESX سرور (میزبان) فیزیکی را به بخش های مجزا، قابل انتقال و کاملاً ایمن تقسیم می کند به طوری که سیستم های عامل در حال اجرا در هر بخش می توانند به راحتی و به طور مستقل در کنار بقیه کار کنند. منابع انتزاعی تخصیص یافته به هر سیستم عامل به عنوان یک ماشین مجازی شناخته می شود. در واقع هر ماشین مجازی امکانات یک کامپیوتر کامل از جمله پردازنده، حافظه، ورودی/خروجی، سرویس شبکه و ذخیره سازی و حتی BIOS را به سیستم عامل مربوط به خود ارائه می دهد. بنابراین سیستم عامل میهمان و نرم افزار های مربوط به آن می توانند به راحتی بر روی این ماشین مجازی نصب شده و کار کنند. همچنین بر خلاف بسیاری دیگر از نرم افزارهای hypervisor، سیستم عامل ها و نرم افزارهای میهمان برای نصب شدن بر روی ESX نیاز به هیچ تغییری ندارند.

بنابراین ESX بر روی یک میزبان از یک سو باید مدیریت منابع سرور را انجام دهد و از سوی دیگر سرویس های لازم را برای ماشین های مجازی فراهم آورد. این سرویس ها شامل دریافت درخواست های آنان برای استفاده از منابع گفته شده و انتقال درخواست مذکور به سخت افزار یا منبع حقیقی می باشد. همچنین پس از اجرا شدن دستور توسط منبع مورد نظر، باید پاسخ به سیستم عامل مطرح کننده درخواست عودت داده شود.

۴,۶,۲. ارائه یک مدل فرمال از نحوه کار ESX

با توجه به جزئیات مطرح شده در فصل ۲، لازم است پس از ارائه یک مدل کلی از ESX، به طرح مدل هایی از هر یک از بخش های اجرایی آن پرداخته می شود. در ادامه مدلی انتزاعی از ESX در شکل ۴,۲۰ آورده شده است.



شکل ۴,۲۰. مدل پتری طرز کار ESX

همانطور که مشاهده می شود جزئیات مربوط به مکانیزم ها و شیوه مدیریت منابع در سرور توسط ESX در این مدل آورده نشده است. بخش های اصلی سیستم از جمله پاسخگویی به درخواست

های شبکه و دیسک را در بخش های بعدی به شکل دقیقتری بررسی خواهیم کرد. در ادامه به تشریح گزارهای مدل می پردازیم.

- T0: درخواست ماشین مجازی برای دسترسی به دیسک
- T1: درخواست ماشین مجازی برای استفاده از سرویس شبکه
- T2: فراخوانی پشته ذخیره سازی¹ برای رسیدگی به درخواست
- T3: درخواست مطح شده برای دسترسی به یک دیسک غیر مبتنی بر شبکه است
- T4: درخواست مطح شده برای دسترسی به یک دیسک مبتنی بر شبکه است
- T5: عملیات داخلی دیسک برای ارائه پاسخ
- T6: پاسخگویی دیسک به پشته ذخیره سازی
- T7: عدم پاسخ مناسب به پشته ذخیره سازی به دلیل بروز خطا در فرایند خواندن/نوشتن از دیسک

- T8: گزارش خطا از سوی پشته ذخیره سازی به ماشین مجازی
- T9: دریافت پاسخ از دیسک مبتنی بر شبکه به پشته ذخیره سازی
- T10: ارائه پاسخ (سرویس) از جانب پشته ذخیره سازی به ماشین مجازی
- T11: طرح درخواست عملیات دیسک تحت شبکه از سوی پشته ذخیره سازی به پشته شبکه²
- T12: عملیات داخلی vSwitch برای انتقال درخواست از شبکه داخلی میزبان به شبکه خارجی
- T13: ارائه درخواست سرویس شبکه از راه انداز کارت شبکه فیزیکی
- T14: عملیات شبکه خارجی
- T15: ارائه پاسخ (سرویس) به ماشین مجازی
- T16: ارائه پاسخ سرویس دیسک به پشته ذخیره سازی
- T17: عدم پاسخ مناسب به پشته شبکه به دلیل بروز خطا در فرایند سرویس شبکه
- T18: گزارش خطا از سوی پشته شبکه به ماشین مجازی

¹ storage stack

² Network stack

همانطور که دیده می شود گزارهای T5، T12 و T14 در این مدل زمان دار هستند به این معنی که اجرای عملیات برای گذر کردن سیستم از این مرحله نیازمند زمان است. در چنین شرایطی می توان نتیجه گرفت که مکانیزم مفصلی در این بخش ها به صورت یک گزار واحد خلاصه شده است. بنابراین فرایند ارائه سرویس شبکه و دیسک و همچنین نحوه دقیق تر کار vSwitch باید بررسی شود.

در این مدل با طی شدن دائمی حلقه درخواست عملیات دیسک و یا سرویس شبکه، چرخه دائمی دریافت درخواست ماشین های مجازی و پاسخگویی به آنها انجام می شود. با توجه به اینکه بخش بزرگی از کار این دو حلقه به محیط خارج از سرور یعنی دیسک و شبکه بستگی دارد، احتمال بروز خطا و استثنا^۱ در آنها زیاد است. اما برای کنترل پیچیدگی در نحوه مدل سازی این نقاط بروز خطا، آنها را در دو نقطه جمع می کنیم. به این معنی که نتیجه مثبت یا منفی نهایی اجرای سرویس را در پایان اجرای عملیات در گزار T7 برای فرایند سرویس دیسک و در گزار T17 برای فرایند سرویس شبکه نشان می دهیم. قرار گرفتن سیستم در وضعیت های P7 و P13 باعث ارائه گزارش خطا به ماشین مجازی و بازگشت سیستم به وضعیت اولیه خواهد شد.

همچنین باید توجه داشت که در صورت مطرح شدن درخواست سرویس دیسک دو وضعیت بروز خواهد کرد. در صورتی که فایل مورد درخواست بر روی دیسکی واقع شده باشد که مبتنی بر شبکه Ethernet است، این درخواست پس از تحلیل توسط پشته ذخیره سازی به سرویس شبکه تحویل داده می شود تا مانند یک درخواست معمولی شبکه با آن برخورد شود. به این معنی که پیشخ شبکه، همانند درخواست های دیگر آن را به مقصد ارسال و جواب آن را دریافت می کند. با این تفاوت که در اینجا درخواست کننده نه یک ماشین مجازی که سرویس دیسک خود ESX بوده است. در پایان نیز پاسخ دریافت شده از شبکه به پشته ذخیره سازی تحویل داده خواهد شد. نمونه هایی از این نوع دیسک را می توان تحت عنوان iSCSI^۲ نام برد. در این مدل، کنترل عملیات پس از وضعیت P5 با

¹ Exception

² Internet SCSI

ارائه درخواستی به پشته شبکه به سرویس شبکه سپرده می شود و در نهایت با اجرای گزار T16 در صورت موفقیت آمیز بودن عملیات دیسک تحت شبکه، سیستم در وضعیت P8 قرار می گیرد. در صورتی که فایل مورد درخواست بر روی یک سیستم ذخیره سازی مانند دیسک داخلی سرور، SCSI معمولی و یا Fiber SCSI باشد، به نحوی که اجرای عملیات نیازی به سرویس شبکه نداشته باشد، پشته ذخیره سازی این کار را به طور مستقل انجام می دهد. در این مدل، با اجرای گزار T3 سیستم در وضعیت اجرای عملیات از دیسک غیر شبکه ای قرار گرفته و تا رسیدن سیستم به مرحله P9 در این پروسه می ماند.

همچنین با طرح درخواست سرویس شبکه چه از سوی یک ماشین مجازی و چه از جانب پشته ذخیره سازی، سیستم در وضعیت P2 قرار می گیرد. در پایان عملیات شبکه خارجی و اجرای گزار T14 سیستم در وضعیت P12 است که به معنی اتمام عملیات شبکه و دریافت پاسخ از کارت شبکه فیزیکی می باشد. در این حالت گزارهای T15، T16 و T17 فعال و آماده اجرا شدن هستند. نوع پاسخ دریافت شده از شبکه تعیین می نماید که کدامیک از این سه گزار باید اجرا شوند. در صورتی که پاسخ دریافت شده مربوط به درخواست یک ماشین مجازی برای استفاده از سرویس شبکه بوده باشد، گزار T15 اجرا شده و پاسخ به ماشین مذکور تحویل داده می شود. در صورتی که طرح کننده درخواست سرویس پشته ذخیره سازی بوده باشد، با اجرای گزار T16 پاسخ به وی ارائه می شود. و در صورتی که پاسخ دریافت شده حاکی از خطا در پردازش درخواست بوده باشد، با اجرای گزار T17 این خطا به پشته شبکه گزارش داده می شود.

در قسمت های بعدی از این بخش به طراحی و تحلیل مدل زیر مکانیزم های ESX از جمله سرویس شبکه و دیسک می پردازیم.

۴,۶,۳. بررسی و تحلیل مدل

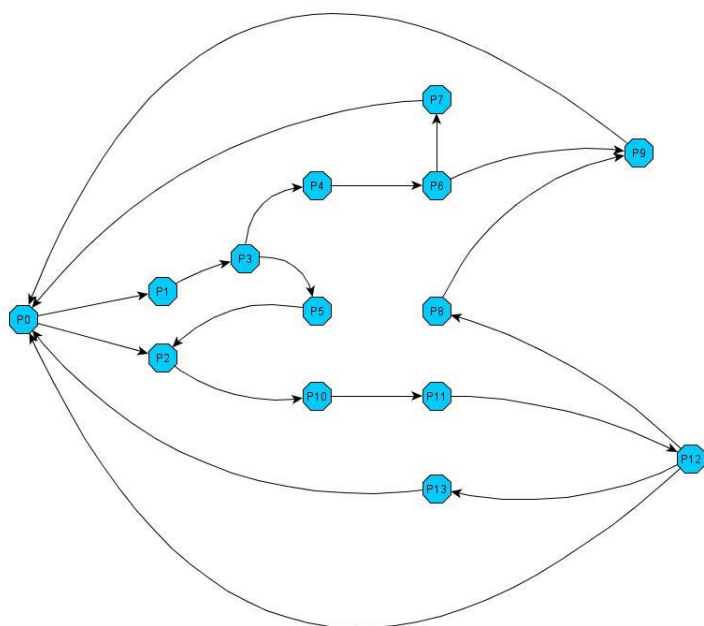
در این بخش با تحلیل خصوصیات رفتاری مدل، میزان انعطاف پذیری و پایداری آن و در نهایت خوش رفتاری سیستم را بررسی خواهیم کرد. به همین منظور خصوصیات **Safeness**، **Liveness** و **Reversibility** را بر روی آن مطالعه می نمائیم.

در ابتدا باید توجه داشت که تمام گزارها در شبکه فوق دارای تنها یک ورودی و یک خروجی هستند. بنابراین این شبکه، یک **state machine** است.

Liveness ۴,۶,۳,۱

بر اساس قضیه ۴ از فصل ۲، یک **state machine** در صورتی دارای خصوصیت **liveness** خواهد بود که به عنوان یک گراف جهت دار، همبند قوی باشد و در وضعیت M_0 ، حداقل یک توکن در آن وجود داشته باشد.

گراف جهت دار $G(V,E)$ در صورتی همبند قوی است که برای هر $u,v \in E$ یک مسیر جهت دار از u به v و نیز یک مسیر جهت دار از v به u وجود داشته باشد. بنابراین باید مدل پتری را به صورت یک گراف جهت دار نشان دهیم. با توجه به اینکه در **state machine** هر گزار دقیقا یک لبه ورودی و یک لبه خروجی دارد، می توان در این مدل از رسم گزارها به عنوان راس صرف نظر کرد تا گراف حاصل ساده تر شود. این گراف در شکل ۴,۲۳ نشان داده شده است.



شکل ۴,۲۱. گراف جهت دار معادل شبکه پتری ۴,۲۰

برای نشان دادن خصوصیت همبندی قوی در این گراف می توان از الگوریتم کزاراجو استفاده کرد. این شیوه که از جستجوی اول عمق^۲ و گراف ترانهاد G^T استفاده می کند، در واقع برای تشخیص بخش های همبند قوی یک گراف^۳ (SCC) معرفی شده است. اگر بتوان نشان داده که در گراف $G(V,E)$ تنها یک SCC وجود دارد، این بدان معنی خواهد بود که کل گراف همبند قوی است. این الگوریتم در ادامه توضیح داده شده است [۷۱].

- ۱- فرض می کنیم G یک گراف جهت دار و S یک پشته خالی است.
- ۲- تا زمانی که S همه گره های گراف را در خود جای نداده است عملیات زیر را انجام می دهیم:

• گره دلخواه v را که در S موجود نیست انتخاب می کنیم. عملیات جستجوی اول عمق را با شروع از v انجام می دهیم. در اجرای این جستجو، هر بار

¹ Kosaraju

² Depth first search

³ strongly connected components

که الگوریتم، توسعه یک گره را در هنگام بازگشت خاتمه داد، گره مذکور را در S ثبت می کنیم.

۳- گراف ترانهاده G را به دست می آوریم. G^T با معکوس کرده جهت همه لبه های G حاصل می شود.

۴- تا زمانی که S خالی نشده:

- عنصر بالای پشته را خوانده (v)، عملیات جستجوی اول عمق را با شروع از v انجام می دهیم. مجموعه گره هایی که در این پیمایش خوانده می شوند در یک زیر گراف همبند قوی با حضور v قرار دارند (SCC). این گره ها را ثبت کرده و آنها را از گراف G و همچنین پشته S حذف می کنیم.

به این ترتیب می توان حوزه های همبند قوی را در یک گراف جهت دار شناسایی نمود.

با اجرای این الگوریتم بر روی گراف ۴,۲۱ می بینیم که همه گره ها عضو یک حوزه همبند قوی بوده و در نتیجه کل گراف همبند قوی می باشد. به این ترتیب شبکه پتری متناظر با آن یعنی مدل ۴,۲۰ نیز همبند قوی خواهد بود.

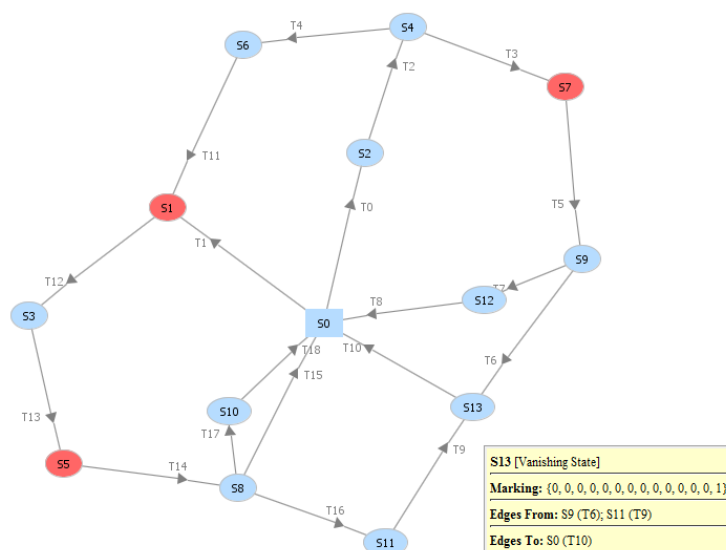
با استناد به قضیه ۴، همبند قوی بودن مدل پتری ۴,۲۰ و نیز حضور یک توکن در شبکه در وضعیت M_0 ، می توان نتیجه گرفت که شبکه مذکور $live$ است.

۴,۶,۳,۲. Safeness

با توجه به قضیه ۵، یک state machine که دارای خصوصیت liveness می باشد، در صورتی safe خواهد بود که در وضعیت M_0 دقیقا یک توکن در شبکه وجود داشته باشد. زیرا با توجه به اینکه وزن تمام لبه ها ۱ است و همه گزارها در شبکه نیز با هر بار اجرا شدن دقیقا یک توکن مصرف کرده و یک توکن ایجاد می کنند، تعداد توکن های قابل انتقال به یک موقعیت، هرگز بیش از ۱ نخواهد بود. پس شبکه 1-bounded یا safe خواهد بود.

در شبکه ۴,۲۰ در M_0 دقیقا یک توکن وجود دارد بنابراین این شبکه safe است.

همچنین با بررسی گراف پوشای مدل ۴,۲۰، خواهیم دید که در هیچ یک از وضعیت های محتمل در گردش کنترل شبکه، تعداد توکن های موجود در موقعیت ها بیش از یک نمی شود. این امر نشان می دهد که تمام موقعیت های شبکه 1-bounded و یا safe هستند. گراف پوشای این شبکه پتری در شکل ۴,۲۲ آمده است.



شکل ۴,۲۲. گراف پوشای مدل پتری ۴,۲۰

۴,۶,۳,۳. Reversibility

برای بررسی این خصوصیت لازم است به تعریف Reversibility توجه کنیم. با یاد آوری تعریف این مفهوم و نیز خاصیت Reachability از فصل ۲ داریم: شبکه پتری (N, M_0) Reversible خوانده می شود اگر برای هر وضعیت M ، در دنباله قابل اجرا از M_0 ، $(R(M_0))$ از طریق M ، reachable باشد.

با این تعریف، در شبکه های کوچک می توان با بررسی کلیه وضعیت های محتمل در شبکه، reachable بودن M_0 را از طریق آنها بررسی نمود. در صورتی که M_0 از طریق تک تک این وضعیت ها (M_1 تا M_n) قابل دسترسی باشد این شبکه Reversible است. با توجه به اینکه این

وضعیت ها تماما در گراف پوشای مدل منعکس می شوند می توان با بررسی یک به یک گره های گراف این حالات را بررسی نمود.

به کمک الگوریتم کزاراجو می توان نشان داد که گراف $4,22$ یک گراف همبند قوی است. این بدان معنی است که از هر گره به گره ی دیگر مسیر جهت داری وجود دارد. در نتیجه از هر گره S_i به S_0 مسیری وجود دارد. پس امکان بازگشت به M_0 از هر M_i در شبکه پتری مربوط به آن وجود دارد.

در نتیجه می توان گفت که شبکه پتری مذکور Reversible است.

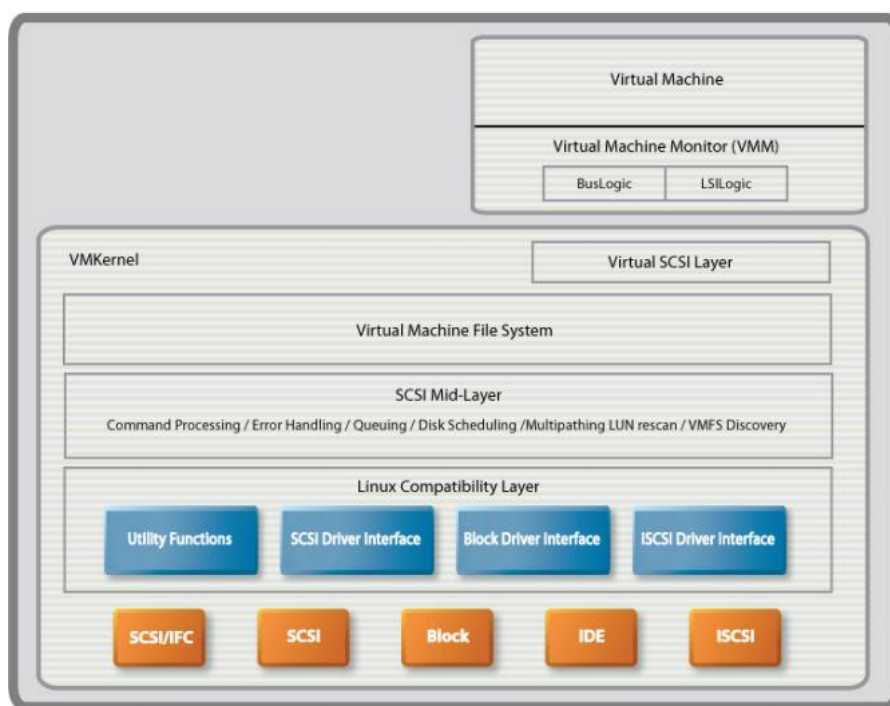
۴,۷. تشریح ساختار سیستم ذخیره سازی در ESX

با تکیه بر مقدمات ارائه شده در فصل ۲، در این بخش به توضیح مفصل تری درباره ساختار ذخیره سازی در ESX خواهیم پرداخت تا در بخش های بعدی بتوانیم مدل فرمالی از آن تهیه کنیم.

۴,۷,۱. تشریح ساختار سیستم ذخیره سازی

در این بخش به بررسی مدل چند لایه ای سیستم ذخیره سازی فایل ها در ESX و نحوه کار آنها خواهیم پرداخت [۶۳] و [۷۳].

با توجه به اینکه در یک دیتا سنتر به احتمال زیاد طیف متنوعی از دستگاه های ذخیره سازی در کنار منابع پردازشی مورد استفاده قرار می گیرد، لازم است این بخش در ESX انعطاف و قابلیت همکاری زیادی با این تجهیزات داشته باشد. شکل ۴,۲۱ شمایی از مدل لایه ای معماری ذخیره سازی در ESX را نشان می دهد.



شکل ۴،۲۳. نمای شماتیک مدل چند لایه ای سیستم ذخیره سازی در ESX [۶۳]

بخش های اصلی در شکل مذکور عبارتند از:

- Virtual Machine Monitor (VMM)
- Virtual SCSI Layer
- VMware File System (VMFS)
- SCSI Mid-Layer
- HBA Device Drivers
- Virtual Machine Monitor (VMM)

Virtual Machine Monitor (VMM)

وظیفه اصلی ماژول VMM مانیتور نمودن کلیه رفتار و فعالیت های ماشین مجازی در تمام سطوح است. این بخش ها شامل پردازنده، حافظه، ورودی / خروجی، و دیگر فعالیت ها و روابط آن با VMkernel می باشد. ماژول VMM شامل لایه ای است که دستگاه های SCSI را برای ماشین های مجازی شبیه سازی می کند. سیستم عامل ماشین مجازی دسترسی مستقیمی به تجهیزات

ذخیره سازی ندارد زیرا زیر ساخت مجازی سازی به کمک ماژول ذخیره سازی، تنها یک رابط SCSI در اختیار این سیستم عامل قرار داده است. بنابراین بدون توجه به نوع سیستم عامل، از هر ماشین مجازی نرم افزارها تنها از طریق یک راه انداز SCSI به دیسک دسترسی دارند. ماشین های مجازی می توانند از راه انداز BusLogic یا LSI برای کار با دیسک های SCSI استفاده کنند. این راه اندازها امکان استفاده از SCSI HBA را برای ماشین های مجازی فراهم می کند.

BusLogic

استفاده از راه انداز BusLogic به این معنی است که شبیه ساز BT-958 در حال استفاده است. این شبیه ساز در واقع نمود پیاده سازی شده پروتکل SCSI-3 است که امکان Ultra SCSI را برای انتقال سریع با نرخ 40MBs فراهم می کند. این شبیه ساز امکان پیکربندی خودکار SCSI را فراهم می کند که به عنوان SCAM¹ شناخته می شود. این امکان به دیسک SCSI امکان می دهد که با یک شماره شناسه به طور خودکار پیکربندی شود. بنابراین مدیر سیستم نیازی به نسبت دادن دستی شناسه نخواهد داشت.

LSI Logic

استفاده از این راه انداز تعیین می کند که سیستم در حال استفاده از شبیه ساز LSI53C1030 می باشد. این شبیه ساز از تکنولوژی SCSI-320 استفاده می کند. از مزایای این تکنولوژی می توان به مصرف کم انرژی، اعتبار سنجی در دامنه SCSI در هنگام عملیات بر روی دیسک، همخوانی با تکنولوژی PCI-X و بهینه سازی در بررسی افزونگی اشاره کرد. در کنار این مزایا، این شبیه ساز از تکنولوژی TolerANT برای تحمل بیشتر نوسانات سیگنال استفاده می کند. از مزایای دیگر این راه انداز می توان به استفاده از منطق نفی فعال² بر روی راه انداز SCSI و فیلتر سازی سیگنال ورودی به دیسک برای بهبود وضعیت تمامیت داده ها اشاره کرد.

¹ SCSI Configured AutoMatically

² Active Negation

تمام مکانیزم ها و مزایای گفته شده برای نرم افزارهایی که از یک SAN برای ذخیره سازی اطلاعات خود استفاده می کنند بسیار مفید است. زیرا این نرم افزارها باید در مقابل مشکلات و اختلالات سیستم ذخیره سازی از جمله قطعی کابل ها، ترکیب بخش های مختلف یک SAN، یا مغایرت بخش های SAN که باعث اختلال در سیستم ذخیره سازی می شود مقاوت داشته باشند. یک مزیت کلیدی دیگر در LSI53C1030 پشتیبانی زیر ساختی از تکنولوژی انتقال پیام ادغامی است که به عنوان Fusion-MPT شناخته می شود. این معماری با فراهم کردن مکانیزم کارآمدی برای پردازنده سیستم میزبان، به کنترل گر ورودی/خروجی در سیستم امکان می دهد تا با ارسال همزمان چندین پیام پاسخ به پردازنده در یک سیگنال وقفه، از تعداد context switch ها به نحو محسوسی بکاهد. این راهکار باعث افزایش نرخ انتقال تا 100000 IOPS بر حسب معیار Ultra320 SCSI می شود. این در حالی است که بار پردازشی سیستم و میزان مداخله تجهیزات جانبی در حداقل میزان ممکن است.

^۲ SCSI HBA مجازی به ماشین های مجازی امکان می دهد که به دستگاه های SCSI منطقی دقیقاً مانند یک HBA فیزیکی متصل شود و به وی اجازه اتصال به دستگاه های ذخیره سازی فیزیکی را می دهد. اما برخلاف HBA فیزیکی، HBA های مجازی به مدیر سیستم اجازه نمی دهد که به ماشین های فیزیکی دسترسی داشته باشند.

بر روی یک سیستم ESX، هر ماشین مجازی می تواند از یک تا چهار SCSI HBA مجازی داشته باشد. این آداپتورهای مجازی می توانند از کنترل کننده های BusLogic و یا LSI استفاده کنند. لازم به ذکر است که این دو راه انداز تنها انتخاب ماشین های مجازی برای کار با دیسک های SCSI است.

Virtual SCSI Layer

¹ Fusion Message Passing Technolog

² host bus adapter

وظیفه اصلی لایه SCSI مجازی مدیریت دستورات SCSI و برقراری ارتباط میان لایه VMM، لایه VMFS و زیر لایه SCSI در زیر خودش می باشد. همه دستورات SCSI صادر شده از طرف ماشین های مجازی باید از این لایه عبور نماید. همچنین ناتمام ماندن عملیات ورودی/ خروجی و تنظیم و اجرای مجدد عملیات در این لایه مورد بررسی قرار می گیرد. از این محل، دستورات ورودی/ خروجی و دستورات SCSI از طرف ماشین های مجازی به لایه های پایینتر ابلاغ می شود. این ابلاغ دستور توسط VMFS و یا RDM صورت می گیرد که اولی برای دسترسی از طریق فایل سیستم VMware و دومی برای دسترسی مستقیم ماشین مجازی به دیسک است. همچنین درخواست عملیات دیسک از طریق RDM به دو صورت pass-through و non-pass-through انجام می شود که در حالت اول دستورات SCSI بدون ایجاد trap به دیسک ابلاغ می شود.

VMware File System

VMFS به طور اختصاصی برای VMware ESX ساخته شده و برای ذخیره و بازیابی فایل های بزرگ بهینه شده است. در این فایل سیستم، با استفاده از بلاک های بزرگ، کارایی دیسک در ماشین های مجازی نزدیک به کارایی آنها در کار با دیسک فیزیکی نگه داشته می شود. VMFS برای سیستم های SAN بسیار مناسب است زیرا از یک منطق داخلی برای اسکن مجدد استفاده می کند که باعث می شود تغییر در LUN ها را به طور خودکار کشف نماید. مزیت کلیدی دیگر این فایل سیستم این است که پیچیدگی سیستم ذخیره سازی فیزیکی را با پنهان نگه داشتن خطاهای SAN از ماشین های مجازی کاملاً پوشیده می سازد. منحصر به فرد ترین خصوصیت VMFS این است که به عنوان یک فایل سیستم کلاستر، منابع ذخیره سازی مشترک را به نحوی مدیریت می کند که به طور همزمان به چندین سرور فیزیکی اجازه نوشتن و خواندن از یک دیسک واحد را می دهد. VMFS از یک قفل توزیع شده بر روی دیسک استفاده می کند تا مطمئن شود که یک ماشین مجازی همزمان توسط بیش از یک سرور فیزیکی اجرا نمی شود. امکان این قفل توسط SCSI-2 فراهم می شود. در صورتی که سرور فیزیکی از کار بیفتد، قفل دیسک برای فایل های همه ماشین های مجازی آن سرور آزاد شده و این ماشین

ها می توانند بر روی سرورهای دیگری اجرا شوند. همچنین VMFS امکاناتی برای مقابله با اختلال و بازیابی اطلاعات در سطح کلان ارائه می دهد که از آن جمله می توان به ژورنالینگ توزیع شده، مسیر ورودی/ خروجی مجازی مقاوم در برابر اختلال^۱ و تصویر برداری از وضعیت ماشین اشاره کرد. این ابزار می تواند در بررسی سریع و بازیابی اطلاعات روی ماشین های مجازی، سرورهای فیزیکی و زیر سیستم های ذخیره سازی کمک نماید.

SCSI Mid-Layer

این لایه در واقع مهمترین لایه در معماری ذخیره سازی VMkernel است و وظایفی از جمله نگاشت HBAهای فیزیکی به میزبانان ESX، مدیریت صف درخواست ها و رسیدگی به خطاهای SCSI را انجام می دهد. همچنین این لایه دارای یک مکانیزم منطقی داخلی برای اسکن کردن LUNهای نگاشته شده به یک ESX و یافتن تغییرات آنها است. مدیریت کانال^۲ از جمله انتخاب اتوماتیک کانال، حذف کانال و رفع ایراد یک بخش از دیسک^۳ در این لایه انجام می شود. لایه SCSI اطلاعات را از HBAها، سوئیچ ها و پردازنده های ذخیره سازی جمع آوری کرده و به کمک آنها کانال لازم را بین میزبان ESX و ولوم فیزیکی بر روی آرایه ذخیره سازی ایجاد می کند. در طول اسکن، ESX به دنبال اطلاعاتی مانند شناسه NAA^۴ و شماره سریال می گردد. ESX تمام کانال های موجود را بررسی کرده و در نهایت بدون توجه به تعداد آنها، یکی را انتخاب می کند. همه کانال های انتخاب نشده به عنوان کانال جانشین در نظر گرفته می شوند.

۴،۷،۲. ارائه یک مدل فرمال از نحوه کار سیستم ذخیره سازی در ESX

در این بخش با تکیه بر جزئیات فنی گفته شده در بخش قبل به ارائه یک مدل فرمال با زبان شبکه های پتری می پردازیم.

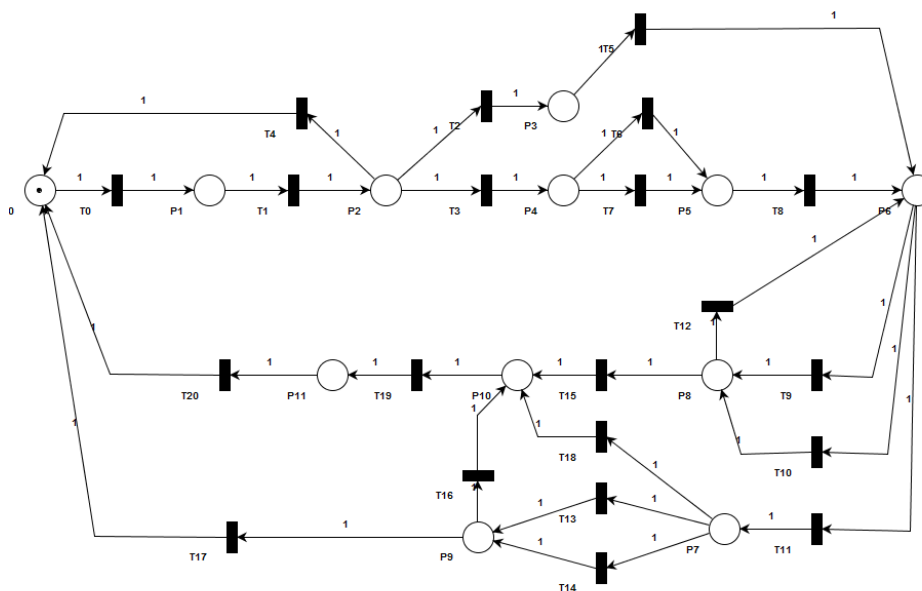
شکل ۴،۲۲ مدل طراحی شده را نشان می دهد.

^۱ crash-consistent virtual machine I/O path

^۲ Path

^۳ Volume

^۴ network address authority



شکل ۴،۲۴. مدل پتری ارائه شده از نحوه کار سیستم ذخیره سازی در ESX

در زیر شرح گزارهای سیستم آورده شده است.

T0: طرح درخواست دسترسی به دیسک از طرف ماشین مجازی

T1: فراخوانی راه انداز مجازی SCSI توسط لایه VMM

T2: فراخوانی ماژول RDM برای دسترسی مستقیم ماشین مجازی به یک LUN

T3: فراخوانی VMF برای دستیابی به دیسک

T4: بروز خطا در ورودی/ خروجی راه انداز SCSI مجازی و فراخوانی تابع reset

T5: فرایند نگاشت فایل توسط RDM برای ارائه سروی خواندن و یا نوشتن

T6: بررسی و رفع ایراد LUN در صورت بروز خطا

T7: قفل نمودن فایل مورد نظر بر روی دیسک توسط VMFS

T8: فرایند باز کردن فایل توسط VMFS

T9: بررسی خودکار برای یافتن تغییرات ایجاد شده در LUN

T10: مدیریت کانال ارتباطی بین VMkernel و ولوم فیزیکی دیسک

T11: فراخوانی راه انداز دیسک فیزیکی

T12: برگرداندن جواب مثبت و ارائه داده ها در صورت خواندن از فایل

T13: ارسال درخواست توسط راه انداز فیزیکی به دیسک

T14: قرار دادن درخواست در صف

T15: کشف خطا در فرایند خواندن و نوشتن از دیسک فیزیکی در مرحله برقراری ارتباط با

دیسک فیزیکی

T16: کشف خطا در فرایند خواندن و نوشتن از دیسک فیزیکی در مرحله دریافت پاسخ از دیسک

فیزیکی

T17: ارائه بازخورد به ماشین مجازی در مورد اجرای درست دستور

T18: کشف خطا در فرایند خواندن و نوشتن از دیسک فیزیکی در مرحله ارسال درخواست به

دیسک فیزیکی

T19: رسیدگی به خطای دیسک فیزیکی

T20: بازخورد به ماشین مجازی در مورد بروز خطا

در بررسی این مدل باید توجه داشت که هر بخشی از کار توسط یکی از لایه ها انجام می شود. با توجه به اینکه درخواست عملیات دیسک از سوی ماشین مجازی صادر شده است، انتظار می رود که این وظایف از لایه های بالا شروع شده و به ترتیب به پایین ختم شود. همچنین انجام بعضی از فرایندها، آنطور که در این مدل در گزارهای مشخص خلاصه شده است، لزوماً بر عهده یک لایه مشخص نمی باشد بلکه ممکن است توسط دولایه به صورت مشترک انجام گیرد.

نکته مهم دیگر که باید در بررسی فرایند عملیات دیسک در محیط مجازی مد نظر داشت، ماهیت خطاپذیر بودن سیستم است. با توجه به اینکه بخشی از عملیات در خارج از ESX و در همکاری با شبکه و سیستم های ذخیره سازی انجام می شود، احتمال بروز اختلال و استثنا به خصوص در خارج از سرور فیزیکی زیاد است. بنابراین باید در بررسی روند تغییر کنترل در سیستم به بروز خطاها توجه

داشت و مکانیزم ها و نقاطی را برای رسیدگی به آنها در نظر گرفت. در غیر این صورت سیستم ناپایدار و غیر قابل اطمینان خواهد بود.

همانطور که در مدل دیده می شود، پس از طرح درخواست ماشین مجازی برای دستیابی به دیسک، بلافاصله راه انداز مجازی SCSI توسط لایه VMM فراخوانی می شود. با رجوع به توضیحات مدل لایه ای سیستم ذخیره سازی خواهیم دید که این راه انداز تنها نقطه دسترسی ماشین مجازی به دیسک است که می تواند از دو تکنولوژی BusLogic یا LSI استفاده کند. در مرحله ای که درخواست ماشین مجازی توسط راه انداز مجازی دریافت شده است سیستم در وضعیت P2 قرار دارد. اگر در این مرحله به هر دلیل راه انداز نتواند درخواست را به لایه های پایینتر منتقل کند، گزار T4 رخ داده و سیستم پس از گزارش خطا و reset کردن راه انداز به وضعیت اول باز می گردد. لازم به ذکر است که این بررسی خطا و ارائه بازخورد به ماشین مجازی در لایه virtual SCSI انجام شده و به لایه VMM منتقل می شود. این حالت می تواند در صورت عدم انتخاب درست راه انداز یا موجود نبودن دیسک SCSI به صورت فیزیکی و غیره رخ دهد.

در صورت تحلیل درست درخواست در لایه VMM، این درخواست به صورت یک فراخوانی به لایه پایینتر یعنی virtual SCSI ارائه می شود. در این لایه که از مهمترین بخش های سیستم است، باید در ابتدا تشخیص داده شود که دیسک مورد فراخوانی به چه شکلی در اختیار ماشین مجازی قرار خواهد گرفت. طبق توضیحات قسمت قبل، با توجه به تنظیمات مدیر سیستم، ماشین مجازی می تواند به دیسک مجازی خود از طریق فایل سیستم VMware (VMFS) دسترسی داشته باشد و یا می تواند به طور مستقیم از طریق سرویس RDM به بخش نگاشت یافته ای از دیسک فیزیکی دسترسی داشته باشد. در حالت اول سیستم با اجرای گزار T3 در وضعیت P4 قرار می گیرد به این معنی که درخواستی به VMFS برای بازیابی فایل مورد نظر بر روی LUN مشخص ارسال می شود. در واقع این درخواست به لایه پایینتر یعنی لایه VMFS ارسال خواهد شد. در حالت دوم، سیستم با اجرای گزار T2 در وضعیت P3 قرار می گیرد. به عبارت دیگر سرویس RDM فراخوانی می گردد.

در صورتی که ماشین مجازی به طور مستقیم به بخشی از دیسک دسترسی داشته باشد و درخواست به مازول RDM ابلاغ شود، این مازول با نگاشت مستقیم، فایل را باز کرده و آماده اجرای عملیات خواندن/نوشتن می شود. به عبارت دیگر با اجرای گزار T5 مستقیماً سیستم را در وضعیت P6 قرار می دهد. در غیر اینصورت درخواست به لایه VMFS ابلاغ شده و وی می تواند دو عمل مجزا را انجام دهد. در حالت طبیعی این لایه باید قفل دیسک را برای فایل مورد نظر فعال کند تا ماشین مجازی یا ESX دیگری نتواند به طور همزمان با آن کار کند. ولی در صورتی که اختلال و مشکلی در کار با فایل یا آن بخش از دیسک که فایل بر روی آن قرار دارد پیش آمده باشد، VMFS باید ابتدا به آن رسیدگی کرده و آن را رفع نماید. در حالت اول گزار T7 و در حالت دوم گزار T6 اجرا می شود تا سیستم را به وضعیت P5 ببرد. در این حالت VMFS آماده است که فایل را برای عملیات خواندن/نوشتن باز نماید.

از مرحله P6 کنترل در اختیار لایه پایینتر یعنی SCSI mid-layer قرار می گیرد که بخش بسیار مهمی از عملیات را به عهده خواهد داشت. با توجه به یک مکانیزم منطقی در داخل این لایه، LUNهای دیسک به طور مداوم اسکن می شوند تا تغییرات آنها ثبت شود. بنابراین با اجرای گزار T9 عملیات اسکن انجام شده و سیستم در وضعیت P8 قرار می گیرد. در صورت موفقیت آمیز بودن عملیات اسکن، با اجرای گزار T12 وضعیت کنترل سیستم دوباره در حالت P6 قرار گرفته و برای اجرای دیگر توابع این لایه آماده می شود.

با توجه به اینکه ایجاد و نگهداری از کانال ارتباطی بین ESX و راه انداز دیسک فیزیکی از وظایف این لایه است، در صورت لزوم، عملیات ایجاد کانال، بررسی کانال های موجود و یا جایگزین کردن یک کانال جانشین به جای کانال فعال (در صورت از بین رفتن کانال فعال) توسط لایه در این مرحله انجام می شود. بنابراین با اجرای گزار T10 سیستم پس از مدیریت کانال ها باز هم به وضعیت P8 منتقل می شود که در صورت موفقیت آمیز بودن عملیات، با اجرای T12، لایه آماده اجرای فرایند جدید خواهد بود.

اگر در هر یک از دو مرحله گفته شده یعنی اسکن کردن LUN مورد نظر و یا مدیریت کانال ارتباطی خطایی رخ دهد، سیستم از وضعیت P8 با اجرای T15 به P10 خواهد رفت. به این معنی که با کشف خطا آماده بررسی و رسیدگی به آن می شود.

در نهایت در صورت در دسترس بودن کانال مناسب و اسکن شدن دیسک، عملیات فراخوانی راه انداز دیسک فیزیکی می تواند توسط لایه انجام شود. در این حالت که معادل اجرای گزار T11 در مدل می باشد، لایه مذکور با دسترسی به راه انداز فیزیکی، درخواست را به وی ابلاغ می کند و سیستم در وضعیت P7 قرار می گیرد. اگر فراخوانی راه انداز به هر دلیل دچار مشکل شده و فرایند کامل نشود، گزار T18 رخ خواهد داد که همانند T15 با کشف خطا سیستم را به P10 می برد. در غیر اینصورت دو حالت محتمل خواهد بود. در حالت اول راه انداز می تواند درخواست را مستقیم به سمت دیسک ارسال نماید و در حالت دوم به دلیل شلوغی خط و یا اختلال موقتی و در دسترس نبودن دیسک امکان طرح بلادرنگ درخواست وجود ندارد بنابراین درخواست در یک صف ذخیره می شود. حالت اول با اجرای گزار T13 و حالت دوم با اجرای T14 در مدل مذکور نشان داده شده است. اجرای هر دو گزار سیستم را به وضعیت P9 خواهد برد.

اگر در اجرای درخواست ها مشکلی بروز کند سیستم با اجرای گزار T16 برای کشف خطا به P10 می رود در غیر این صورت پاسخ دریافت شده از دیسک از مسیر طی شده برای ماشین مجازی ارسال خواهد شد.

در صورتی که در هر یک از مراحل گفته شده، سیستم با کشف خطا به وضعیت P10 برسد، با اجرای مازول های رسیدگی و کنترل خطا مشکل بررسی و طی گزار T20 به لایه های بالاتر و در نهایت ماشین مجازی گزارش داده می شود.

در قسمت بعدی به بررسی و تحلیل مدل ارائه شده خواهیم پرداخت.

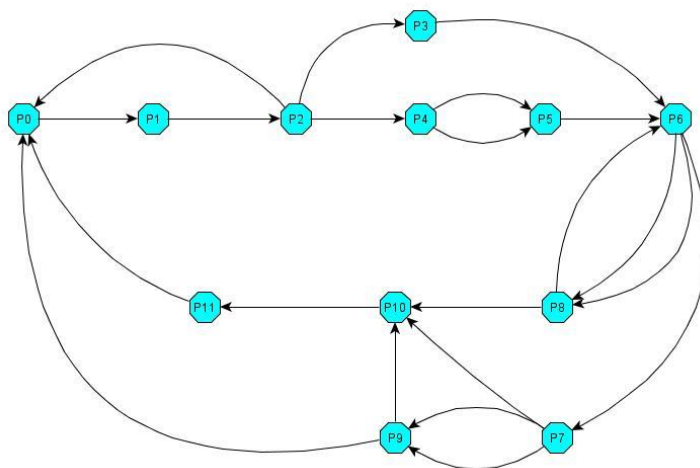
۴,۷,۳. بررسی و تحلیل مدل

در این بخش با تحلیل خصوصیات رفتاری مدل، میزان انعطاف پذیری و پایداری آن و در نهایت خوش رفتاری سیستم را بررسی خواهیم کرد. به همین منظور خصوصیات *Safeness*, *Liveness* و *Reversibility* را بر روی آن بررسی می نمائیم.

در ابتدا باید توجه داشت که تمام گزارها در شبکه فوق دارای تنها یک ورودی و یک خروجی هستند. بنابراین این شبکه، یک *state machine* است.

۴,۷,۳,۱. *Liveness*

طبق قضایای فصل ۲، یک *state machine* در صورتی *live* است که همبند قوی باشد و M_0 حداقل یک توکن داشته باشد. گراف جهت دار معادل مدل ۴,۲۴ در شکل ۴,۲۵ آمده است. برای ساده تر شدن گراف همه گزارها در آن به صورت لبه های ارتباطی بین موقعیت ها در نظر گرفته شده اند. این عمل به دلیل تعریف *state machine* که در آن هر گزار فقط یک ورودی و یک خروجی دارد ممکن است.



شکل ۴,۲۵. گراف جهت دار متناظر با مدل پتری ۴,۲۲

برای نشان دادن خصوصیت همبندی قوی در این گراف از الگوریتم کزاراجو استفاده می کنیم. با اجرای این الگوریتم بر روی گراف ۴,۲۵ می بینیم که همه گره ها عضو یک حوزه همبند قوی بوده و در نتیجه کل گراف همبند قوی می باشد. به این ترتیب شبکه پتری متناظر با آن یعنی مدل ۴,۲۴ نیز همبند قوی خواهد بود.

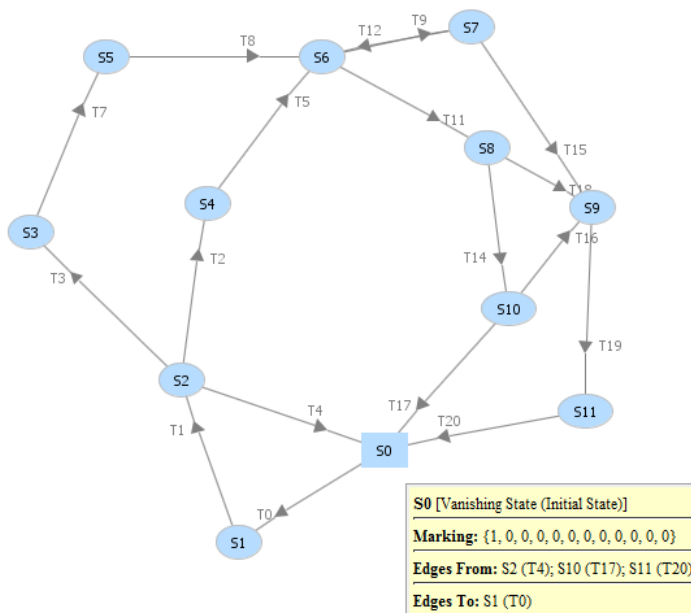
با استناد به قضیه ۴، همبند قوی بودن مدل پتری ۴,۲۴ و نیز حضور یک توکن در شبکه در وضعیت M_0 ، نشان دهنده وجود خصوصیت liveness در شبکه پتری مذکور است.

۴,۷,۳,۲. Safeness

با توجه به قضیه ۵، یک state machine که دارای خصوصیت liveness می باشد، در صورتی safe خواهد بود که در وضعیت M_0 دقیقا یک توکن در شبکه وجود داشته باشد. زیرا با توجه به اینکه وزن تمام لبه ها ۱ است و همه گزارها در شبکه نیز با هر بار اجرا شدن دقیقا یک توکن مصرف کرده و یک توکن ایجاد می کنند، تعداد توکن های قابل انتقال به یک موقعیت، هرگز بیش از ۱ نخواهد بود. پس شبکه 1-bounded یا safe خواهد بود.

در شبکه ۴,۲۴ در M_0 دقیقا یک توکن وجود دارد بنابراین این شبکه safe است. برای بررسی بیشتر می توان نگاهی به گراف پوشای شبکه مذکور که منعکس کننده تمام حالات قابل وقوع در آن است داشت. در این گراف که در بخش های قبلی نیز برای بررسی خصوصیات شبکه های پتری از آن کمک گرفته شده است، هر گره نشان دهنده یک وضعیت M_i است که با برچسبی از تعداد توکن های مجموعه موقعیت ها شبکه نشان داده می شود. به عنوان مثال در شکل ۴,۲۶ بر حسب وضعیت M_0 (S_0) به صورت یک توکن در P_0 و هیچ توکن در بقیه موقعیت ها نشان داده شده است.

با بررسی تمام این یازده وضعیت محتمل در شبکه خواهیم دید که هیچ گاه تعداد توکن ها در یک موقعیت بیش از ۱ نبوده است. به این ترتیب هرگز بیش از یک توکن در یک موقعیت قرار نخواهند گرفت.



Marking corresponds to {P0, P1, P10, P11, P2, P3, P4, P5, P6, P7, P8, P9}

شکل ۴,۲۶. گراف پوشای شبکه پتری ۴,۲۴

Reversibility ۴,۷,۳,۳

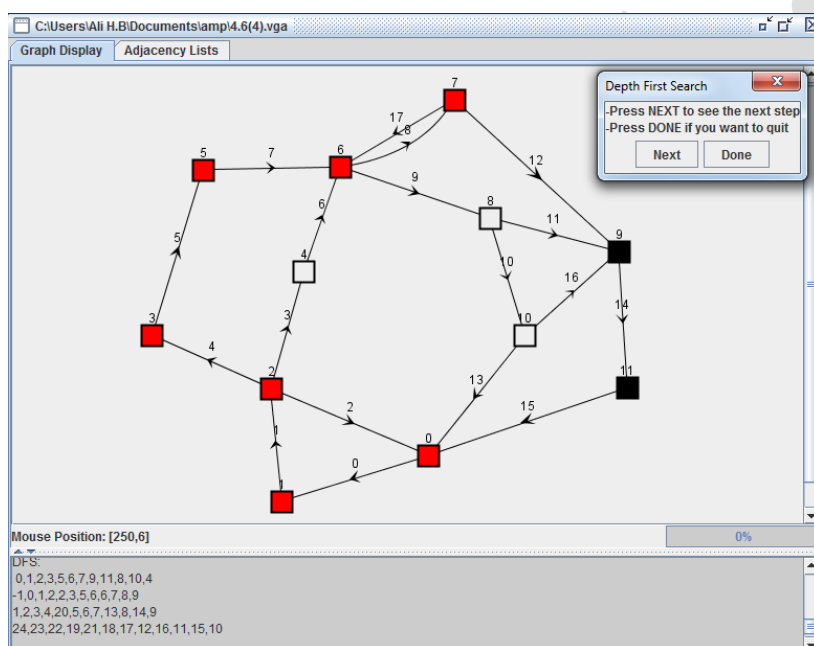
برای بررسی این خصوصیت لازم است به تعریف Reversibility توجه کنیم. با یاد آوری تعریف این مفهوم و نیز خاصیت Reachability از فصل ۲ داریم: شبکه پتری (N, M_0) Reversible خوانده می شود اگر برای هر وضعیت M ، در دنباله قابل اجرا از M_0 ، $(R(M_0))$ از طریق M reachable باشد.

با این تعریف، در شبکه های کوچک می توان با بررسی کلیه وضعیت های محتمل در شبکه، reachable بودن M_0 را از طریق آنها بررسی نمود. در صورتی که M_0 از طریق تک تک این وضعیت ها (M_1 تا M_n) قابل دسترسی باشد این شبکه Reversible است. با توجه به اینکه این وضعیت ها تماماً در گراف پوشای مدل منعکس می شوند می توان با بررسی یک به یک گره های گراف این حالات را بررسی نمود.

اگر نشان دهیم که گراف ۴,۲۶ یک گراف همبند قوی است، آنگاه مطمئن خواهیم بود که از هر گره به گره ی دیگر مسیر جهت داری وجود دارد. این بدان معنی است که از هر گره S_i به S_0

مسیری وجود دارد. بنابراین با استفاده از الگوریتم کزاراجو این گراف را برای یافتن حوزه های همبند قوی بررسی می کنیم.

به کمک اجرای دستی الگوریتم و جستجوی اول عمق به وسیله نرم افزار تحلیلگر گراف با نام “Virtual Graph Analyzer”، می بینیم که تنها یک حوزه همبند قوی در این گراف وجود دارد که شامل تمام گره های آن است. بنابراین این گراف همبند قوی است. (شکل ۴،۲۷)



شکل ۴،۲۷. نرم افزار تحلیلگر گراف، در حال اجرای الگوریتم اول عمق

به این ترتیب می توان نتیجه گرفت که از هر گره ی این گراف (هر وضعیت M_i) مسیری برای رسیدن به M_0 وجود دارد. به عبارت دیگر از هر M_i با اجرای یک دنباله متناهی از گزارها می توان به M_0 رسید. بنابراین مدل پتری ۴،۲۴ Reversible است.

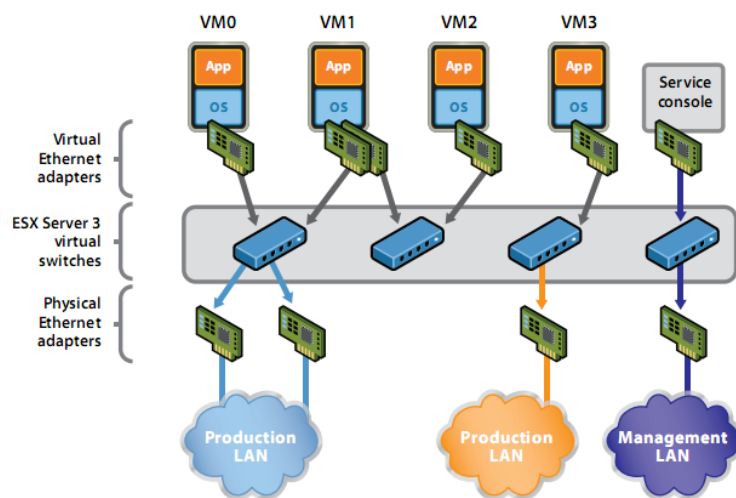
۴,۸. معماری ساختار شبکه در ESX

در ادامه تشریح ساختار ESX و تحلیل مدل های مربوط به این ساختارها، در این بخش با تکیه بر مقدمات ارائه شده در فصل ۲، به تشریح دقیقتر ساختار شبکه در ESX hypervisor می پردازیم تا در بخش بعد بتوانیم یک مدل فرمال به کمک شبکه های پتری از آن ارائه نمائیم. لازم به یادآوری است که طبق بخش بندی ارائه شده در ابتدای فصل، در این بخش ما لایه سوم از مدل کلی دیتا سنترها را ارائه خواهیم کرد. همچنین در این مسیر لازم خواهد بود بخش هایی از مدل ساختار شبکه شامل مکانیزم vSwitch به طور جداگانه مدل شود که یک مدل لایه چهارم را تشکیل خواهد داد.

۴,۸,۱. تشریح ساختار شبکه

مدل لایه ای شبکه در ESX، ساختار ماژولار و انعطاف پذیری را برای کار با انواع سخت افزارهای شبکه و سیستم عامل ها فراهم می آورد. بخش کلیدی که توسط ساختار شبکه در محیط مجازی ارائه شده و مورد استفاده قرار می گیرد، مفهوم کارت شبکه مجازی^۱ است. این ماژول توسط ماشین های مجازی و سوئیچ مجازی مورد استفاده قرار می گیرد. سوئیچ مجازی نرم افزاری در ساختار شبکه ESX است که ارتباط بین ماشین های مجازی با یکدیگر و نیز ارتباط آنها با شبکه خارج از سرور ESX را فراهم می کند. شمای کلی این نحوه همکاری در شکل ۴,۲۸ نشان داده شده است.

^۱ virtual Ethernet adapter



شکل ۴,۲۸. نحوه ارتباط کارت شبکه مجازی و سوئیچ مجازی [۶۲]

در ادامه به بررسی دقیقتر هر یک از بخش های این ساختار چند لایه و نحوه کار آنها می پردازیم [۶۲]، [۷۳] و [۷۴].

نحوه کار کارت شبکه مجازی

این مازول در واقع رابط نرم افزاری ماشین مجازی با شبکه خارجی است. به طور کلی سه نوع کارت شبکه برای یک ماشین مجازی قابل استفاده است که ماشین می تواند در آن واحد تنها از یک نوع آن استفاده کند. این انواع عبارتند از:

• vmxnet

یک رابط شبکه نیمه مجازی^۱ است که تنها در صورتی کار می کند که نرم افزار مکمل VMware Tools بر روی سیستم عامل میهمان نصب شده باشد. عبارت نیمه مجازی در مورد این کارت شبکه به این معنی است که این مازول از اجرا شدن در یک محیط مجازی مطلع است و

^۱ Paravirtualized

بنابراین کاملاً مستقل از ESX نمی باشد. کارت شبکه مجازی vmxnet برای کار با کیفیت بسیار زیاد ساخته شده است.

در صورتی که مدیر سیستم نرم افزار VMware Tools را بر روی سیستم عامل میهمان نصب کرده باشد، با انتخاب اتوماتیک کارت شبکه مجازی، این نوع کارت شبکه برای سیستم برگزیده می شود.

• vlance

این نوع کارت شبکه در واقع شبیه سازی دقیقی از کارت مدل AMD Lance PCNet32 فراهم می کند که با اکثر سیستم عامل های ۳۲ بیتی سازگار است. در صورتی که نرم افزار VMware Tools بر روی سیستم عامل میهمان نصب نشده باشد، با انتخاب اتوماتیک کارت شبکه، این نوع کارت برای ماشین مجازی در نظر گرفته می شود.

• e1000

شبیه سازی دقیقی از کارت شبکه E1000 است که توانایی کار در محیط های ۶۴ بیتی را دارد. البته این آداپتور با سیستم عامل های ۳۲ بیتی نیز سازگار است.

همه کارت های شبکه نام برده شده در بالا، خصوصیات زیر را به طور مشترک دارند:

- دارای آدرس MAC مخصوص به خود هستند و فیلترهای unicast، multicast و broadcast را دارند.

- دقیقاً دستگاه های کارت شبکه در لایه ۲ هستند که به صورت نرم افزاری پیاده سازی شده اند.

لازم به تذکر است که مسائلی چون سرعت و duplex که در کارت های شبکه فیزیکی وجود دارد در مورد کارت های مجازی شبکه مطرح نیست. زیرا تمام فرایند انتقال ترافیک در داخل سیستم

¹ Emulation

میزبان و بر روی حافظه RAM در حال انجام است که تقریباً بدون تاخیر انجام می شود. همچنین در چنین شرایطی، برخورد بسته ها در لایه فیزیکی^۱ و دیگر خطاهای سیگنال وجود ندارد.

نحوه کار سوئیچ مجازی^۲

این دستگاه بخش کلیدی ساختار شبکه در ماشین های مجازی است. مدیر سیستم می تواند حداکثر ۲۴۸ سوئیچ در هر ESX ایجاد نماید. به طور کلی کارکردهای این سوئیچ را می توان به صورت زیر برشمرد:

- ابزاری برای forwarding در لایه ۲

این یک بخش کلیدی برای افزایش کارایی و کارکرد صحیح سیستم در زیر ساخت مجازی است و برای ساده سازی بیشتر، تنها وظایف لایه ۲ را انجام می دهد. در واقع یک سوئیچ مجازی، کاملاً مستقل از تفاوت های دیگر اجزای مجاور خود از جمله کارت های شبکه فیزیکی و مجازی پیاده سازی شده است.

- کار کردن با VLANها از جمله برچسب گذاری، خواندن برچسب ها عملیات

فیلترسازی بر حسب شماره VLAN

- اعمال امنیت در لایه ۲، checksum و اعمال segmentation offload

این ساختار ماژولار در شبکه، به سیستم اجازه می دهد زمانی که سوئیچ مجازی شروع به کار می کند، سرور ESX تنها بخش هایی را که برای کار سوئیچ ضروری هستند بارگزاری کرده و اجرا نماید. این اجزا می توانند شامل راه اندازهای کارت شبکه های فیزیکی و مجازی و یا دیگر ماژول های وابسته به آنها باشند. بنابراین سیستم از نظر پیچیدگی و میزان تقاضای منابع، کمترین هزینه را برای ارائه سرویس شبکه در داخل سرور فیزیکی می پردازد.

به دلیل ماهیت ماژولار بودن سیستم عامل ESX، این سیستم می تواند به طور موقتی ماژول هایی را بارگزاری و از آنها استفاده کند و در صورت عدم نیاز، آنها را از حافظه کاری حذف نماید. این

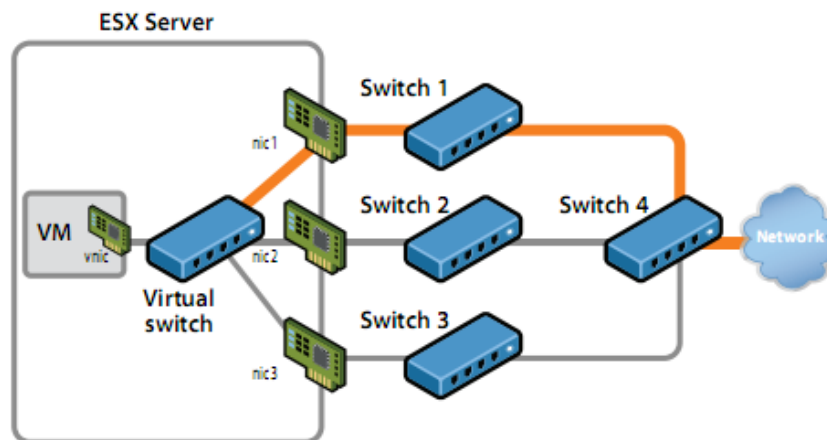
¹ Collision

² vSwitch

ساختار مشابه شیوه کار هسته لینوکس و بسیاری دیگر از سیستم عامل های POSIX است که بر حسب نیاز ماژول هایی را بر روی هسته اضافه و یا کم می کنند. در مورد ESX این توانایی در مواقعی چون استفاده از ابزار تشخیص خطا و یا مانیتورینگ موقت بسیار مفید است. از دیگر فواید این ساختار ماژولار در معماری شبکه ESX، می توان به توانمند ساختن شرکت ها و گروه های توسعه دهنده سخت افزار و نرم افزار در افزودن بخش هایی به این سیستم برای بهبود کارایی و یا استفاده های خاص منظور اشاره کرد. به دلیل اهمیت این بخش از ساختار شبکه، سوئیچ های مجازی را در بخش بعد به طور دقیقتری بررسی و مدل خواهیم کرد.

Port Group

این مفهوم به معنی ایجاد یک الگوی ثابت برای ایجاد پورت های مجازی است. باید توجه داشت که چنین مفهومی در سوئیچ های فیزیکی وجود ندارد و مختص سوئیچ های مجازی است. مدیر سیستم می تواند حداکثر ۵۱۲ گروه برای پورت های مجازی بر روی یک ESX ایجاد نماید. با توجه به اینکه به کمک گروه بندی پورت ها می توان تعداد زیادی پورت کاملاً مشابه ساخت، این امکان برای استفاده از سرویس هایی مانند VMotion که در آنها باید مشخصات کارت های شبکه در ماشین های مجازی مبدا و مقصد یکسان باشد بسیار مفید است. همچنین باید توجه داشت که مفهوم port group کاملاً مستقل از VLAN بوده و لزوماً ارتباط یک به یکی بین این گروه ها و گروه های VLAN وجود ندارد. در زیر شمایی از نحوه قرار گیری کارت شبکه های مجازی و سوئیچ مجازی در مقابل شبکه فیزیکی را مشاهده می کنید.



شکل ۴,۲۹. شمای کلی از ساختار شبکه در سرور ESX [۶۲]

نقش VLAN ها در ساختار شبکه ESX

تکنولوژی شبکه های محلی مجازی یا VLAN ابزاری برای گروه بندی سیستم های متصل به شبکه یا به عبارت بهتر گروه بندی پورت های متصل به آنها در سوئیچ است. با این ابزار می توان یک شبکه فیزیکی را به طور نرم افزاری به چندین شبکه کوچکتر تقسیم کرد بدون اینکه نیاز به قرار دادن سوئیچ های متعدد باشد. با این کار سیاست های دقیقی در مورد نحوه دسترسی این شبکه ها به یکدیگر و جریان ترافیک بین آنها قابل اعمال است. همچنین با محدود کردن دامنه broadcast به درون هر VLAN به جای کل شبکه، حجم زیادی از ترافیک زائد در شبکه حذف خواهد شد. شیوه کار در شبکه هایی که با VLAN از یکدیگر جدا شده اند به این صورت است که سوئیچ با دریافت هر بسته از پورت، شبکه مجازی که پورت مذکور در آن قرار گرفته را چک می کند. سپس شماره VLAN مذکور را در سرایند لایه ۲ بسته ثبت می نماید. در انتها نیز سوئیچ پس از مشخص کردن پورت مقصد بسته به کمک جدول forward، شماره VLAN بسته را چک می کند. در صورتی که پورت مقصد در همان VLAN باشد به بسته اجازه انتقال داده می شود و در غیر اینصورت بسته حذف می گردد.

این عملیات برچسب گذاری و بررسی برچسب ها در مبدا و مقصد توسط پروتکل های متعددی قابل انجام است که دو نمونه از پرکاربرد ترین آنها IEEE 802.1Q و ISL است. پروتکل اول توسط سازمان IEEE و پروتکل دوم توسط شرکت Cisco ایجاد شده اند.

برای اینکه امکان پشتیبانی از VLAN در شبکه مجازی وجود داشته باشد، لازم است بخشی در این ساختار مانند کارت شبکه مجازی، سوئیچ مجازی و یا کارت شبکه فیزیکی سرور این وظیفه را بر عهده گیرد. در اینجا سه وضعیت پیکربندی برای استفاده از امکان VLAN در ساختار شبکه ESX وجود دارد که در ادامه به تشریح آن می پردازیم.

- برچسب گذاری سوئیچ مجازی^۱ (VST)

در این حالت که متداول ترین حالت تنظیم VLAN در سیستم مجازی می باشد، مدیر سیستم به ازای هر VLAN یک port group بر روی سوئیچ مجازی تعریف کرده و VLAN را به آن نسبت می دهد. سپس به جای اینکه کارت های شبکه مجازی متعلق به هر ماشین مجازی را مستقیماً به سوئیچ متصل کند، آنها را به port group مذکور نسبت می دهد. این port group که در واقع الگویی برای تعریف پورت های سوئیچ مجازی است با قرار گرفتن در یک VLAN مشخص، عمل برچسب گذاری بر روی بسته های خروجی و خواندن برچسب بسته های ورودی را انجام می دهد.

در این حالت port group از انتقال بسته های یک VLAN به دیگر VLAN ها جلوگیری می کند. بنابراین مساله ایزوله بودن VLAN مذکور نیز تامین می گردد. باید توجه داشت که برای استفاده از این حالت در ESX باید پورت سوئیچ فیزیکی که کارت شبکه فیزیکی سرور به آن متصل شده در وضعیت trunk قرار گیرد. قرار دادن یک پورت در وضعیت trunk باعث می شود تا پورت مذکور بتواند بسته های مربوط به تمام VLAN ها را از خود عبور دهد.

- برچسب گذاری ماشین مجازی^۲ (VGT)

¹ Virtual switch tagging

² Virtual machine guest tagging

با توجه به اینکه رسیدگی به سرایند مربوط به VLAN در ساختار شبکه به معنی برچسب گذاری و خواند آنها بر روی بسته های ورودی و خروجی است، مدیر سیستم می تواند با نصب یک راه انداز 802.1Q VLAN trunking بر روی سیستم عامل ماشین مجازی، این کار را به کارت شبکه مجازی آن واگذار کند. در این حالت بسته های حاوی برچسب VLAN بین پشته شبکه ماشین مجازی و کارت شبکه فیزیکی سرور تبادل خواهند شد و در این میان سوئیچ مجازی تنها وظیفه forward بسته ها را برعهده خواهد داشت.

برای استفاده از این حالت مانند حالت قبل لازم است پورت متناظر در سوئیچ فیزیکی در حالت trunk قرار داشته باشد.

- برچسب گذاری سوئیچ خارجی (EST)

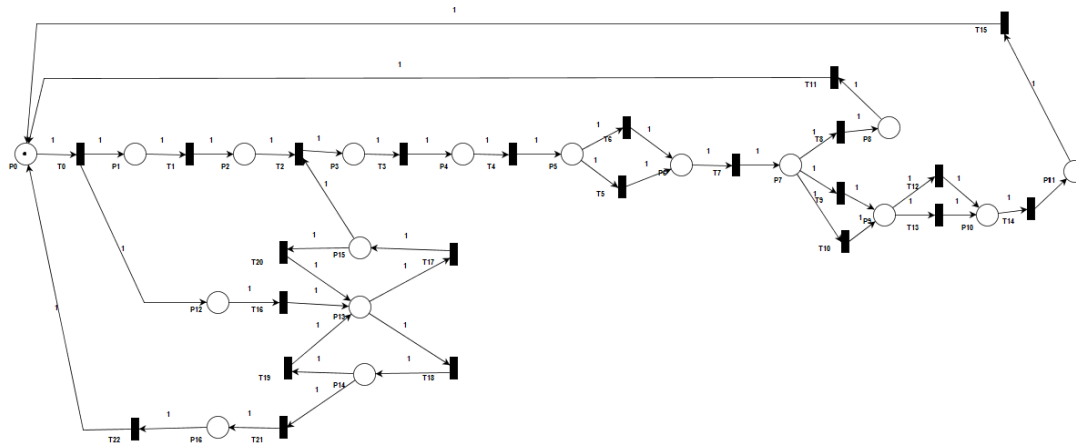
در این حالت وظیفه برچسب گذاری و خواندن آنها به سوئیچ فیزیکی که یکی از پورت های آن به کارت شبکه فیزیکی سرور متصل است واگذار می شود. این دقیقا مانند شرایطی است که یک سرور عادی به شبکه فیزیکی متصل می شود و سوئیچ بر اساس شماره VLAN که سرور در آن قرار دارد به بسته ها اجازه عبور می دهد. در این حالت پورت سوئیچ فیزیکی در وضعیت access و نه trunk قرار می گیرد.

۴,۸,۲. ارائه مدل فرمال برای ساختار شبکه در ESX

در این بخش به بررسی رفتار سیستم شبکه در معماری ESX بر روی مدل فرمال تهیه شده از این سیستم می پردازیم.

یک نمونه از مدل پتری تهیه شده از این سیستم در شکل ۴,۳۰ نشان داده شده است.

¹ External switch tagging



شکل ۴,۳۰. مدل پتری تهیه شده از ساختار شبکه در ESX

در ادامه شرح گزارهای مدل که وضعیت سیستم را تغییر می دهند آمده است.

T0: ارائه درخواست سرویس شبکه توسط سیستم عامل میهمان

T1: فراخوانی راه انداز کارت شبکه مجازی در ماشین مجازی

T2: فراخوانی سرویس vSwitch برای پردازش بسته و اعمال قوانین port group

T3: اعمال قوانین SPAN و port mirroring

T4: اعلام تمام شدن فرایند SPAN

T5: اعمال تغییرات مربوط به VLAN توسط سوئیچ مجازی بر روی بسته در صورتی که پورت

مورد نظر در حالت access باشد (از جمله برچسب گذاری)

T6: اعمال تغییرات مربوط به VLAN توسط سوئیچ مجازی بر روی بسته در صورتی که پورت

مورد نظر در حالت trunk باشد.

T7: اعمال فرایند switching توسط سوئیچ مجازی بر روی بسته

T8: ارسال بسته به صورت unicast به سمت مقصد. در این حالت مقصد به همان سوئیچ مجازی

متصل است. به عبارت دیگر مقصد بسته یک ماشین مجازی دیگر بر روی همان سرور ESX است.

T9: ارسال بسته به صورت unicast به سمت مقصد در حالتی که مقصد در خارج از سرور ESX

قرار دارد.

- T10: ارسال بسته به صورت multicast و یا broadcast به سمت خارج از سرور ESX
- T11: ارسال بازخور به VMkernel در مورد ارسال بسته به سمت مقصد
- T12: ارسال بسته به uplink سوئیچ مجازی در وضعیت بدون NIC-team
- T13: ارسال بسته به uplink سوئیچ مجازی در وضعیت NIC-team همراه با اعمال سیاست teaming گزاری
- T14: فراخوانی راه انداز کارت شبکه فیزیکی سرور
- T15: ارسال بازخور به VMkernel در مورد ارسال بسته به خارج از سرور
- T16: بارگزاری ماژول های لازم برای اجرای سوئیچ مجازی و تست سوئیچ
- T17: اعلام آماده بودن سوئیچ مجازی برای کار
- T18: اعلام اختلال در سوئیچ مجازی
- T19: فرایند رفع ایراد سوئیچ
- T20: بررسی میزان دسترسی پذیری سوئیچ
- T21: عدم امکان رفع ایراد در سوئیچ مورد نظر
- T22: ارسال بازخورد به VMkernel در مورد مشکل سوئیچ و reset شدن سیستم

این مدل نیز مانند مدل سیستم ذخیره سازی ارائه شده در بخش قبل، در واقع یک مدل لایه ۳ از مدل کلی دیتا سنتر ارائه شده در ابتدای فصل است.

در اینجا با فراخوانی سرویس شبکه توسط سیستم عامل ماشین مجازی، پروسه شروع به اجرا شدن می کند. در واقع با دریافت این درخواست و اجرا شدن T0 دو پروسه به طور موازی شروع می شود. در روال اول، با قرار گیری سیستم در وضعیت P1 روند فراخوانی ماژول های لازم برای ارسال بسته به شبکه انجام می شود و در پروسه دوم فرایند آماده سازی و تست سوئیچ مجازی انجام می گردد.

فراخوانی اول با ارسال یک درخواست به راه انداز کارت شبکه مجازی که می تواند یکی از انواع vlance، vmxnet و یا e1000 باشد انجام می شود. در این حالت با اجرای T1 ماشین مجازی

بسته را تولید کرده و آماده ارسال آن به شبکه داخلی سرور ESX می گردد. در اینجا بسته پس از ارسال به سوئیچ مجازی خواهد رسید بنابراین لازم است سوئیچ مجازی آماده دریافت و پردازش بسته باشد. بنابراین پروسه دوم باید تمام شده باشد.

روال دوم، که با قرارگیری توکن در P12 آماده اجرا است، توسط T16 اجرا می شود. این گزار در واقع مرحله بارگزاری ماژول های نرم افزار لازم برای کار سوئیچ و تست نهایی سوئیچ است. در صورتی که این فرایند موفقیت آمیز باشد، سیستم از وضعیت P13 به T17 رفته و آماده بودن سیستم اعلام می شود. در اینجا با قرارگیری در وضعیت P15 آمادگی سیستم نشان داده می شود. از سوی دیگر اگر سیستم موفق به آماده کردن سوئیچ برای پردازش بسته های خارج شده از ماشین مجازی نشود با اجرای گزار T18 سیستم به وضعیت خطا در سوئیچ مجازی رفته و در P14 قرار می گیرد. در این حالت با تلاش سیستم برای رفع خطای سوئیچ دو حالت محتمل است: در حالت اول مشکل برطرف شده و گزار T19 وضعیت سیستم را دوباره به P13 بر می گرداند که در این حالت سیستم دوباره آمادگی سوئیچ را برای پذیرش مسئولیت بررسی خواهد کرد. در حالت دوم سیستم موفق به رفع ایراد نرم افزاری سوئیچ نشده که با اجرای گزار T21 سیستم آماده اعلام بروز خرابی در سوئیچ و برگرداندن وضعیت پشته شبکه به حالت اول می شود. این حالت ممکن است به دلیل تنظیمات نادرست در بخش شبکه و یا از کار افتادن موقتی یکی از ماژول های نرم افزاری مربوط به پشته شبکه در ماشین مجازی و یا ESX رخ دهد.

در صورتی که سیستم به وضعیت P16 دچار نشود سوئیچ آماده سرویس دادن به ماشین مجازی خواهد بود. به این معنی که در نهایت با قرار گرفتن توکن در P15 آمادگی سوئیچ اعلام می گردد. با توجه به اینکه در روال اول پس از فراخوانی راه انداز کارت شبکه سیستم مجازی، بسته آماده خروج از ماشین مجازی است، در واقع یک توکن نیز در P2 قرار دارد. پس گزار T2 آماده اجرا شدن است. اجرای T2 به معنی دریافت بسته توسط سوئیچ مجازی و پردازش آن در لایه ۲ می باشد. این مرحله شامل اعمال سیاست های port group از یک سو و سیاست های امنیتی از سوی دیگر است.

پس از این مرحله و قبل از اینکه بسته از سوئیچ خارج شود لازم است پردازش هایی مانند SPAN بر روی آن انجام شود. در صورت تنظیم این امکان توسط مدیر، در این مرحله یک کپی از بسته به پورت مورد نظر برای مقاصد مانیتورینگ ترافیک ارسال می شود. با اجرای T3 این فرایند انجام شده و با اجرای T4 پایان اجرای آن به سیستم اعلام می شود. بنابراین با قرار گیری سیستم در وضعیت P5 سیستم آماده برچسب گذاری VLAN بر روی بسته است.

با توجه به حالات گفته شده برای پردازش پروتکل VLAN، در صورتی که این وظیفه به سوئیچ واگذار شده باشد در این مرحله بر اساس اینکه پورت دریافت کننده بسته در کدام VLAN باشد عمل برچسب گذاری انجام می شود. همچنین اگر پورت در وضعیت trunk یا VLAN پیش فرض^۱ باشد برچسبی روی آن زده نمی شود. همچنین اگر این فرایند به کارت شبکه مجازی ماشین مجازی و یا سوئیچ فیزیکی واگذار شده باشد این مرحله بدون تعیین برچسب پشت سر گذاشته خواهد شد. سپس در گزار T7 فرایند switching بسته بر اساس آدرس MAC و جدول forward انجام می شود.

با قرار گرفتن سیستم در وضعیت P7 سوئیچ مجازی آماده ارسال بسته به خارج است. در این موقعیت سه حالت ممکن است رخ دهد. در حالت اول و بر اساس فرایند switching انجام شده در مرحله قبل، مقصد بسته یک ماشین مجازی بر روی همان سرور ESX است. در اینجا بسته بدون خارج شدن از سرور توسط سوئیچ مجازی به پورت مجازی مقصد ارسال می شود. در این مدل با اجرا شدن T8 سیستم در وضعیت P8 قرار گرفته و تنها کار باقی مانده ارسال یک بازخورد در مورد پایان کار به VMkernel است که با اجرای T11 این کار نشان داده شده است. پس از آن سیستم به وضعیت اولیه برای پردازش درخواست های دیگر بر می گردد.

در صورتی که مقصد بسته در خارج از سرور مذکور بوده و ارسال به صورت unicast باشد حالت دوم رخ خواهد داد. این حالتی است که سوئیچ مجازی بسته را به خارج از سرور و به سمت کارت شبکه فیزیکی ارسال می نماید. با اجرای گزار T9 سیستم در وضعیت P9 قرار گرفته و آماده ارسال بسته به خارج می شود.

¹ Native VLAN

در نهایت در صورتی که نحوه ارسال بسته به صورت multicast و یا broadcast باشد، حالت سوم اتفاق می افتد که در آن، بسته تنها به پورت های uplink سوئیچ ارسال خواهد شد. این یکی از خاصیت های منحصر به فرد سوئیچ های مجازی در ساختار VMware است که در آن ترافیک پخش (broadcast, multicast) تنها به سمت خارج از سوئیچ ارسال شده و به دیگر پورت های مجازی همان سوئیچ فرستاده نمی شود.

بنابراین در پایان این مرحله در صورتی که لازم باشد ترافیک به خارج از سرور ESX فرستاده شود، سیستم در وضعیت P9 قرار خواهد گرفت. پس از این مرحله VMkernel در مورد اینکه کارت های فیزیکی شبکه باید به صورت team و یا غیر از آن استفاده شوند تصمیم گیری نماید. در صورت تنظیم شدن کارت ها به صورت team، مکانیزم خاصی برای ارسال بسته بر روی هر دو کارت به صورت همزمان به کار می رود که این مکانیزم شبیه برخورد دیگر سیستم عامل ها با کارت های team شده است. همچنین اگر چنین تنظیمی بر روی کارت ها انجام نشده باشد، بر اساس نحوه اتصال uplink سوئیچ مجازی به کارت های فیزیکی، کارت مورد نظر برای ارسال بسته انتخاب می شود. در مدل متناظر، در حالت اول گزار T12 و در حالت دوم گزار T13 اجرا می شود. در هر دو حالت پس از اجرای فرایند سیستم در وضعیت P10 قرار می گیرد که به معنی آماده بودن VMkernel برای فراخوانی راه انداز کارت شبکه فیزیکی است. با فراخوانی این راه انداز و اجرای گزار T11 بسته به راه انداز تحویل داده می شود تا توسط کارت به شبکه خارجی ارسال شود. در این مرحله و پس از ارسال بسته، لازم است با اجرای T15 بازخورد مربوط به مرحله انجام شده به VMkernel داده شود و سیستم به وضعیت اولیه برای دریافت درخواست های دیگر از پشته شبکه ماشین های مجازی برگردد.

در قسمت بعد سعی خواهیم کرد مدل ارائه شده در این بخش را تحلیل نمائیم.

۴,۸,۳. بررسی و تحلیل مدل

در این بخش با تحلیل خصوصیات رفتاری مدل، میزان انعطاف پذیری و پایداری آن و در نهایت خوش رفتاری سیستم را بررسی خواهیم کرد. به همین منظور خصوصیات *Safeness*, *Liveness* و *Reversibility* را بر روی آن بررسی می نمائیم.

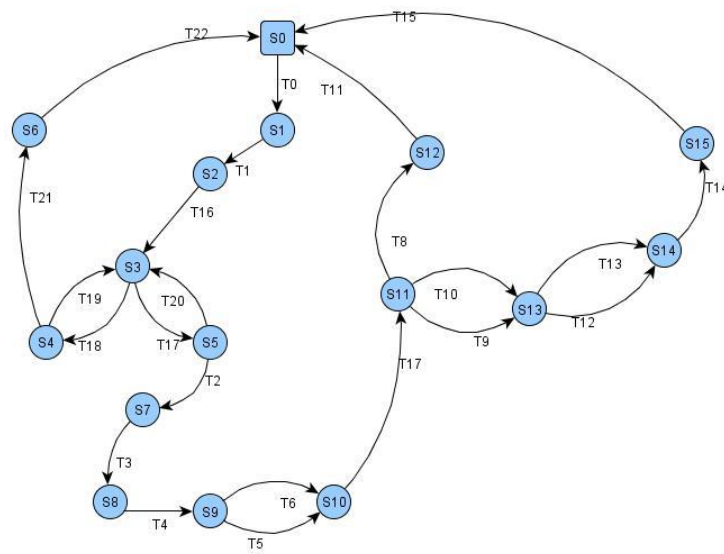
به دلیل اینکه شبکه ۴,۳۰ عضو هیچ یک از زیر مجموعه های شبکه های پتری از جمله *state* *machine*, *marked graph*, *free choice net* و *simple net* نمی باشد، نمی توان از قضایای بیان شده در مورد خواص آنها استفاده نمود. در ادامه با استفاده از روش های قابل اجرا بر روی شبکه های پتری کلاسیک برای مطالعه خصوصیات رفتاری لازم استفاده می کنیم.

۴,۸,۳,۱. *Reversibility*

در این بخش استشنا بررسی *reversibility* در ابتدا آورده شده است زیرا برای بررسی خصوصیت *liveness* به آن نیازمندیم.

همانند قبل برای بررسی این خصوصیت به تعریف *Reversibility* رجوع می کنیم. با یاد آوری تعریف این مفهوم و نیز خاصیت *Reachability* از فصل ۲ داریم: شبکه پتری (N, M_0) , *Reversible* خوانده می شود اگر برای هر وضعیت M , در دنباله قابل اجرا از $M_0, (R(M_0)) M_0$ از طریق M , *reachable* باشد.

با این تعریف، در شبکه های کوچک می توان با بررسی کلیه وضعیت های محتمل در شبکه، *reachable* بودن M_0 را از طریق آنها بررسی نمود. در صورتی که M_0 از طریق تک تک این وضعیت ها (M_1 تا M_n) قابل دسترسی باشد این شبکه *Reversible* است. با توجه به اینکه این وضعیت ها تماما در گراف پوشای مدل منعکس می شوند می توان با بررسی یک به یک گره های گراف این حالات را بررسی نمود. گراف پوشای مدل پتری در شکل ۴,۳۱ آمده است.



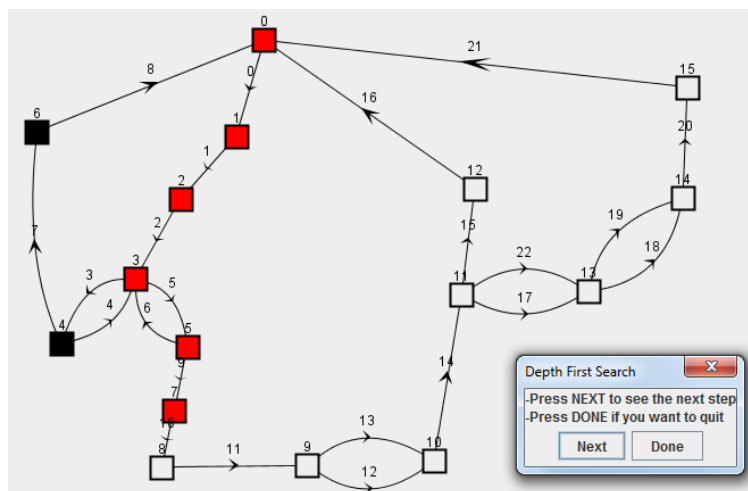
شکل ۴,۳۱. گراف پوشای مدل پتری شکل ۴,۳۰

اگر نشان دهیم که گراف ۴,۳۱ یک گراف همبند قوی است، آنگاه مطمئن خواهیم بود که از هر گره به گره ی دیگر مسیر جهت داری وجود دارد. این بدان معنی است که از هر گره S_i به S_0 مسیری وجود دارد. بنابراین با استفاده از الگوریتم کزاراجو این گراف را برای یافتن حوزه های همبند قوی بررسی می کنیم.

نتیجه جستجوی اول عمق در گراف پوشا به صورت زیر خواهد بود:

$$\text{DFS}(G, S_0) = \{0, 1, 2, 3, 4, 6, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$$

و با اجرای الگوریتم کزاراجو بر روی گراف خواهیم دید که کل گراف یک حوزه همبند قوی است. با این ترتیب می توان مطمئن بود که شبکه پتری متناظر با آن Reversible است. زیرا همیشه مسیری از هر M_i (S_i) به M_0 وجود دارد.



شکل ۴,۳۲. جستجوی اول عمق گراف شکل ۴,۳۱

Liveness ۴,۸,۳,۲

همانطور که گفته شد شبکه پتری مربوط به این بخش (شکل ۴,۳۰) دارای خواص هیچ یک از زیرکلاس های شبکه های پتری نیست. بنابراین برای بررسی liveness در شبکه مربوطه باید به قضایا و لم های کلاسیک در این حوزه رجوع کرد. برای این کار از لم ۱ در فصل ۲ استفاده می کنیم. به کمک این لم می توان بر روی گراف پوشای یک مدل پتری به طور کلی در مورد میزان liveness نظر داد. بر اساس این لم، در صورتی که یک گزار به صورت برچسب یک لبه در گراف پوشا ظاهر نگردد، این گزار dead است. با بررسی گراف پوشای مدل که در شکل ۴,۳۱ آمده است، می بینیم که همه گزارها به صورت لبه ظاهر شده اند پس می توان نتیجه گرفت که گزار dead در مدل مذکور وجود ندارد.

با این حال باید توجه داشت که این به معنی live بودن کل شبکه نیست چرا که ممکن است دنباله ای از گزارها وجود داشته باشد که با اجرای آن شبکه در وضعیت بن بست قرار گیرد. به عنوان مثال با وجود یک گزار sink، با اینکه خود گزار dead نیست، ولی با مصرف توکن، شبکه را از وجود توکن تهی کرده و دیگر امکان اجرای هیچ گزار وجود نخواهد داشت.

به همین دلیل، لازم است با بررسی تک تک گزارهای شبکه سطح liveness آنها را تشخیص داده و در نهایت با توجه به آن، میزان liveness کلی سیستم را مشخص نمود. در ادامه سطوح liveness گزارها را بررسی می کنیم.

با یادآوری مفهوم $L(M_0)$ از فصل ۲، در مدل پتری این بخش، الفبای زبان L تمام وضعیت های ممکن در مدل است که معادل M_0 تا M_{15} در گراف پوشای مدل می باشد. همچنین L شامل تمام دنباله هایی است که M ها با اجرا کردن گزارهای شبکه می توانند ایجاد کنند. یعنی:

$$L(M_0) = \{M_0 M_1 M_2 M_3 M_4 M_6, M_0 M_1 M_2 M_3 M_4 M_3 M_4 M_6, \dots\}$$

بنابراین می توان دنباله های نامتناهی از اجرای M ها ایجاد کرد که همگی عضو L باشند. همچنین باید توجه داشت که قرار گرفتن سیستم در هر یک از وضعیت های M_0 تا M_{15} باعث فعال شدن گزارها طبق جدول ۴،۱ خواهد شد.

جدول ۴،۱. گزارهای فعال در وضعیت های M_0 الی M_{15} از مدل پتری ۴،۳۰

M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
T0	T1	T16	T17, T18	T19, T21	T2, T20	T22	T3	T4	T5, T6	T7	T8, T9, T10	T11	T12, T13	T14	T15

در چنین مجموعه ای، تعداد تکرار هر گزار، درجه liveness آن را تعیین می کند. به این معنی که اگر گزاری در هیچ یک از وضعیت های M_i اجرا نشود dead محسوب می شود. اگر حداقل یک بار در یکی از وضعیت های ممکن اجرا شود live از درجه ۱ (Level-1 live) خواهد بود. همچنین اگر با در نظر گرفتن عدد صحیح و مثبت k ، یک گزار بتواند حداقل k بار در یک دنباله موجود در $L(M_0)$ اجرا شود، Level-2 live خواهد بود. اگر یک گزار بتواند به تعداد نامتناهی در یک رشته از $L(M_0)$ اجرا شود آنگاه Level-3 live خواهد بود. و سرانجام اگر گزاری بتواند در تمام وضعیت های M_i در شبکه اجرا شود Level-4 live می باشد.

با توجه به اینکه شبکه مذکور reversible است، $L(M_0)$ می تواند رشته های نامتناهی از وضعیت ها را با بازگشت مجدد به M_0 ایجاد کند. در چنین رشته هایی هر یک از گزارههایی که به صورت برچسب در گراف پوشا ظاهر شده اند می توانند به تعداد نامتناهی اجرا شوند. با این ترتیب همه گزاره های موجود در گراف Level-3 live خواهند بود. و در نهایت می توان نتیجه گرفت که مدل پتری ۴,۳۰ دارای خصوصیت liveness از درجه ۳ می باشد.

۴,۸,۳,۳. Safeness

با توجه به لم ۱ داریم: شبکه (N, M_0) safe است اگر و تنها اگر فقط مقادیر ۰ و ۱ در برچسب گره های گراف پوشا دیده شود. به این ترتیب چون در برچسب گره های گراف پوشا فقط ۰ و ۱ وجود دارد، می توان نتیجه گرفت که شبکه مذکور دارای خصوصیت Safeness می باشد.

۴,۹. معماری سوئیچ مجازی در ساختار شبکه

در این بخش با بررسی دقیقتر ساختار شبکه در ESX، به تحلیل و ارائه یک مدل فرمال از ساختار سوئیچ مجازی می پردازیم. بررسی این مکانیزم از این بابت حائز اهمیت است که بخش اعظم مسئولیت مدیریت ترافیک شبکه در داخل سرور ESX را بر عهده دارد. بنابراین با تحلیل آن می توان به جنبه های بسیاری از نحوه انتقال کنترل در معماری شبکه ESX پرداخت [۶۲] و [۷۳].

۴,۹,۱. تحلیل ساختار سوئیچ مجازی

در یک نگاه کلی، سوئیچ مجازی از بسیاری جهات شبیه یک سوئیچ لایه ۲ فیزیکی است و از بسیاری جنبه ها نیز با آن متفاوت است. در ادامه به مقایسه این دو نوع سوئیچ می پردازیم تا تحلیل دقیقتری از جایگاه هریک در ساختار شبکه در سیستم های مجازی داشته باشیم.

شباهت های سوئیچ مجازی و فیزیکی

سوئیچ مجازی مانند یک سوئیچ معمولی دارای جدول MAC و جدول port forwarding بوده و اعمال زیر را انجام می دهد:

- آدرس MAC مقصد بسته ها را پس از دریافت بررسی می کند.
- بسته ها را بر حسب آدرس مقصد برای یک یا چند پورت ارسال می کند.
- بسته ها را به صورت نقطه به نقطه و نه broadcast ارسال می کند. به عبارت دیگر وظیفه یک دستگاه لایه ۲ را انجام می دهد و نه یک hub را.

همچنین سوئیچ مجازی توانایی کار با تکنولوژی VLAN را نیز دارد. به عبارت دیگر با قرار دادن هر پورت این سوئیچ در یک VLAN خاص و قرار دادن سوئیچ در حالت access، این پورت تنها توانایی دریافت بسته های مربوط به VLAN خود را دارد. باید توجه داشت که هر پورت یک سوئیچ مجازی به یک کارت شبکه مجازی در یک ماشین مجازی متصل شده است. سوئیچ مجازی این کار را با برچسب گذاری بسته ها در لایه ۲ انجام می دهد همانگونه که یک سوئیچ فیزیکی این کار را انجام می دهد.

همچنین پورت های متصل از یک سوئیچ مجازی به سوئیچ فیزیکی را می توان مانند دو سوئیچ فیزیکی در وضعیت trunk قرار داد تا بتوانند ترافیک همه VLAN را با یکدیگر تبادل کنند.

تفاوت سوئیچ مجازی و فیزیکی

سرور ESX یک کانال مجازی از کارت های شبکه مجازی به سوئیچ مجازی ایجاد می کند که اطلاعات MAC کارت مذکور را در خود دارد. بنابراین در این وضعیت نیازی به اجرای فرایند IGMP snooping برای یادگیری آدرس multicast در شبکه ندارد. همچنین در صورت فعال بودن امکان promiscuous بر روی کارت شبکه مجازی، پورت های متصل به آنها بر روی سوئیچ می توانند بدون هیچ تنظیمی در حالت mirror قرار گیرند. البته برای انجام این کار مجوز سیاست گذاری port group نیز لازم است.

یکی از مهمترین تفاوت های این دو سوئیچ در نحوه اتصال آنها به یکدیگر نهفته است. با توجه به اینکه سیستم مجازی سازی VMware یک ساختار یک لایه ای^۱ را در داخل سرور ESX ایجاد می کند، هیچ راهی برای متصل کردن مستقیم سوئیچ های مجازی در درون یک سرور و یا بین چند سرور ESX وجود ندارد. به همین دلیل هرگز یک حلقه آبین سوئیچ های مجازی به وجود نمی آید. بر این اساس نیازی نیز به استفاده از پروتکل جلوگیری از بروز حلقه (STP) نخواهد بود. البته کاربر می تواند با ایجاد یک bridge مابین دو کارت شبکه مجازی در داخل یک ماشین مجازی، یک حلقه در داخل سوئیچ ایجاد کند. اما این امر به طور تصادفی اتفاق نمی افتد تا نیاز به پیشگیری داشته باشد.

خاصیت ایزوله بودن سوئیچ های مجازی

همانطور که گفته شد، سوئیچ ها هیچگونه ارتباط مستقیمی با یکدیگر در داخل ESX ندارند و به عبارت دیگر ایزوله هستند. این خاصیت مزایایی دارد که عبارتند از: به دلیل عدم نیاز به اتصال سلسله مراتبی سوئیچ ها، با محدود کردن امکان اتصال، ساختار درونی شبکه در ESX بسیار ساده تر می شود. همچنین در این شرایط احتمال اتصال نادرست و خراب سوئیچ ها با یکدیگر به صفر می رسد. و در آخر با توجه به اینکه هر سوئیچ تنها جدول port forwarding مربوط به پورت های خودش را نگه می دارد، یک بسته نمی تواند وارد سوئیچی شود در حالی که به پورتی در سوئیچ دیگر اشاره می کند. با این ترتیب، بسته ها نمی توانند به کارت های شبکه ای که بر روی سوئیچ نیست ارسال شوند و از یک ماشین مجازی به ماشین مجازی دیگر از طریق شبکه نفوذ نمایند.

پورت های مجازی

^۱ Single tier

^۲ Loop

^۳ Spanning Tree Protocol

پورت مجازی بر روی سوئیچ مجازی در واقع یک ارتباط منطقی بین کارت های شبکه ماشین های مجازی و سوئیچ مجازی و همچنین بین سوئیچ مجازی و کارت های شبکه فیزیکی است. می توان این پورت ها را به صورت پورت های RJ-45 معمولی تصور کرد که ارتباطات درون سرور را برقرار می کنند. باید توجه داشت که یک سوئیچ می تواند حداکثر ۱۰۱۶ پورت مجازی داشته باشد. همچنین بر روی یک سرور ESX می تواند حداکثر ۴۰۹۶ عدد از این پورت ها داشت.

پورت های Uplink

این پورت ها در سوئیچ مجازی پورت هایی هستند که ارتباط سوئیچ را با کارت شبکه فیزیکی سرور ESX برقرار می کنند. در واقع سوئیچ مجازی از طریق این پورتها با شبکه فیزیکی ارتباط برقرار کرده و ترافیک کارت های شبکه مجازی موجود بر روی ماشین های مجازی را با شبکه فیزیکی تبادل می کند.

در صورتی که طبق پیکربندی مدیر سیستم، پورت uplink برای سوئیچ مجازی تعریف نشده باشد، کارت های شبکه مجازی که به آن متصل هستند امکان اتصال به شبکه خارجی را نخواهند داشت و فقط می توانند با یکدیگر بر روی همان سوئیچ تبادل بسته نمایند. این طراحی ممکن است در جایی که ترافیک کل این سوئیچ قرار است از یک سیستم ثانویه مانند یک firewall عبور کند مفید باشد.

میزان صحت کارکرد سوئیچ مجازی

به طور کلی در بررسی میزان سلامت کار یک سوئیچ مجازی، بررسی دو نکته حائز اهمیت است که در ادامه به بررسی آنها خواهیم پرداخت. این مساله از آن جهت بسیار مهم است که مطمئن باشیم ماشین های مجازی و دیگر عناصر فعال در شبکه نمی توانند بر روی کار سوئیچ تاثیر بگذارند. در این زمینه ESX حفاظت های لازم را از سوئیچ به عمل می آورد.

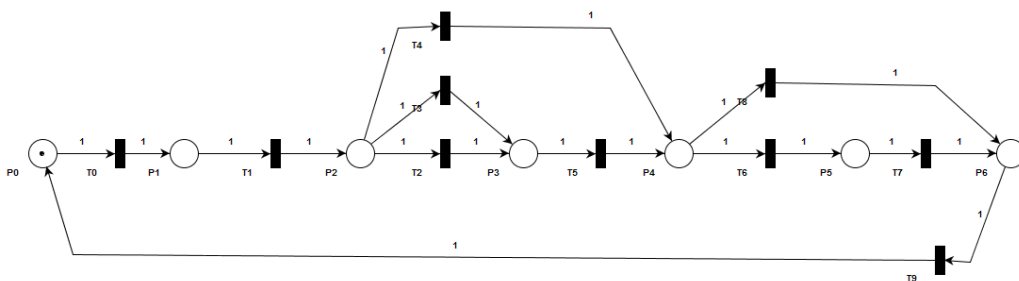
- سوئیچ مجازی نمی تواند آدرس های MAC موجود در شبکه را یاد بگیرد و به این ترتیب جدول forwarding خود را نیز به اشتراک نمی گذارد. این عمل احتمال حملات

¹ DoS و نیز حمله رخنه^۲ را از بین می برد. این حملات ممکن است به صورت مستقیم و عمدی به سرور ESX صورت گیرد که بر روی سوئیچ مجازی تاثیر می گذارد. همچنین این نوع حملات ممکن است تاثیر جانبی بدافزارها و ویروس هایی باشد که به دنبال حفره های امنیتی در سرور می گردند.

- سوئیچ مجازی برای پردازش هر بسته بخش داده ای آن را در جایی کپی می کند و سپس بر روی آن عملیات فیلتر سازی و forwarding را انجام می دهد. این یکی از ویژگی های منحصر به فرد سوئیچ مجازی است که برای تامین امنیت داده های در حال پردازش توسط آن بسیار حیاتی است. باید توجه داشت که کپی کردن کل یک بسته در یک بافر موقتی کاری زمان بر و ناکارآمد است بنابراین سوئیچ تنها بخش داده ای بسته را کپی می کند. این کار به این دلیل انجام می شود که ESX مطمئن شود سیستم عامل های میهمان به داده بسته هایی که در حال انتقال از سوئیچ مجازی است دسترسی نخواهند داشت.

۴,۹,۲. ارائه یک مدل فرمال برای سوئیچ مجازی

در این بخش به ارائه و تشریح مدل پتری متناظر با سوئیچ مجازی شرح داده شده می پردازیم. شمایی از این مدل در شکل ۴,۳۳ دیده می شود.



¹ Denial- of-service

² leakage attacks

شکل ۴,۳۳ مدل فرمال از نحوه کار سوئیچ مجازی

در ادامه شرح گزارهای این مدل که وضعیت کنترل در سیستم را تغییر می دهند آمده است.

T0: بخش داده ای بسته دریافت شده توسط سوئیچ در بافر کپی می شود

T1: سیاست های port group بر روی بسته اعمال می شود

T2: برچسب VLAN توسط سوئیچ مجازی بر روی بسته الصاق می شود

T3: برچسب VLAN قبلا توسط سیستم عامل ماشین مجازی بر روی بسته الصاق شده است

T4: برچسب VLAN بعدا توسط سوئیچ فیزیکی بر روی بسته الصاق خواهد شد.

T5: فرایند تصمیم گیری در مورد پورت مقصد انجام می شود (forwarding)

T6: فرایند SPAN انجام می شود

T7: بسته به پورت مقصد ارسال می شود.

T8: بدون انجام عمل SPAN بسته به پورت مقصد ارسال می شود

T9: ارسال بازخورد به VMkernel در مورد ارسال بسته

در مقابل مدل های توضیح داده شده در بخش های قبل، این یک مدل لایه چهارم از مدل کلی دیتا سنتر می باشد. در واقع این پائینترین سطح مدل سازی است که در این تحقیق به آن پرداخته خواهد شد اما این بدان معنی نیست که امکان تهیه مدل دقیقتر و جزئی تری از شیوه کار سیستم وجود ندارد. بلکه ما می توانیم سیستم را تا سطح اتمیک تحلیل و مدل نمائیم.

فرایند کار سوئیچ مجازی در این مدل، مجموعه ای از گزارها در مدل کلی شبکه (شکل ۴,۳۰) را به شکلی دقیقتر نشان می دهد. بنابراین این مدل شکل دقیقتر تنها یک گزار نمی باشد. می توان شروع آن را پس از اجرا شدن T2 در مدل شکل ۴,۳۰ دانست بنابراین موقعیت P0 در این مدل معادل P3 در شکل ۴,۳۰ است.

قرار گیری توکن در P0 به این معنی است که بسته ای توسط سوئیچ مجازی دریافت شده و بنابراین کار این سیستم شروع می شود. بنابر مطالب گفته شده در قسمت قبل، در ابتدا بخش داده ای بسته توسط سوئیچ به یک بافر موقت کپی می شود. یعنی با اجرای گزار T0 سیستم در موقعیت

P1 قرار می گیرد. در مرحله بعد یعنی اجرای T1 سیاست گزاری های port group مربوط به پورتهای که بسته از آن دریافت شده است، بررسی می گردد.

با قرار گیری سیستم در موقعیت P2 لازم است سوئیچ در مورد نحوه برچسب گذاری بر روی بسته تصمیم گیری نماید. طبق انواع روش های اعمال VLAN، در صورتی که این وظیفه به سوئیچ واگذار شده باشد، گزار T2 برای این برچسب گذاری اجرا می شود. در غیر این صورت بسته به اینکه این وظیفه قبلاً توسط سیستم عامل میهمان انجام شده و یا بعداً توسط یک سوئیچ فیزیکی انجام خواهد شد، گزارهای T3 یا T4 اجرا خواهند شد.

در حالات اول و دوم یعنی در صورتی که برچسب گذاری در درون سرور انجام شود، پس از این مرحله طی گزار T5 عمل تصمیم گیری در مورد پورت مقصد توسط سوئیچ انجام خواهد شد. سپس سیستم در موقعیت P4 قرار می گیرد. در صورتی که وظیفه VLANing به سوئیچ فیزیکی واگذار شده باشد، بدون طی این مرحله سیستم مستقیماً به P4 خواهد رفت. در این حالت طبیعتاً بسته تنها به پورت trunk که از طریق کارت شبکه فیزیکی به سوئیچ فیزیکی متصل است ارسال خواهد شد.

در گام بعد اگر پروتکل SPAN بر روی سوئیچ تنظیم شده باشد، سیستم با اجرای T6 به P5 خواهد رفت و در غیر اینصورت بسته به پورت مقصد ارسال خواهد شد. در صورتی که سیستم در P5 قرار گیرد با اجرای T7 و پس از ارسال یک کپی از بسته به پورت تعیین شده در پروتکل SPAN، بسته به پورت مقصد اصلی ارسال می گردد. در هر دو حالت سیستم پس از ارسال بسته در موقعیت P6 قرار می گیرد. در اینجا تنها لازم است با ارسال بازخورد به VMkernel ارسال بسته را به وی اطلاع دهد و به وضعیت اولیه برگردد.

۴,۹,۳. بررسی و تحلیل مدل

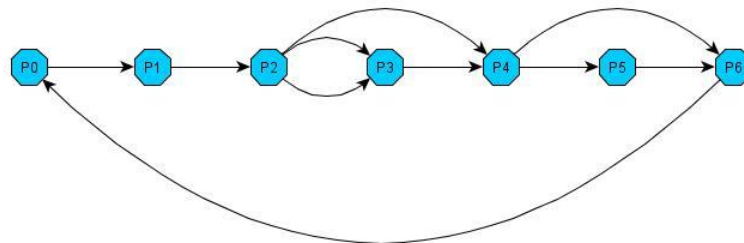
در این بخش با تحلیل خصوصیات رفتاری مدل میزان انعطاف پذیری و پایداری آن و در نهایت خوش رفتاری سیستم را بررسی خواهیم کرد. به همین منظور خصوصیات Safeness, Liveness و Reversibility را بر روی آن بررسی می نمائیم.

در مدل ۴,۳۳ همه گزارها دارای تنها یک لبه ورودی و یک لبه خروجی هستند. این خاصیت، شبکه را در گروه state machine قرار می دهد. بنابراین می توان از قضایای مربوط به این زیرمجموعه برای بررسی خصوصیات مورد بحث استفاده کرد.

Liveness ۴,۹,۳,۱

بر اساس قضیه ۴ از فصل ۲، یک state machine در صورتی دارای خصوصیت liveness خواهد بود که به عنوان یک گراف جهت دار، همبند قوی باشد و در وضعیت M_0 حداقل یک توکن در آن وجود داشته باشد.

گراف جهت دار $G(V,E)$ در صورتی همبند قوی است که برای هر $u,v \in E$ یک مسیر جهت دار از u به v و نیز یک مسیر جهت دار از v به u وجود داشته باشد. بنابراین باید مدل پتری را به صورت یک گراف جهت دار نشان دهیم. با توجه به اینکه در state machine هر گزار دقیقا یک لبه ورودی و یک لبه خروجی دارد، می توان در این مدل از رسم گزارها به عنوان راس صرف نظر کرد تا گراف حاصل ساده تر شود. این گراف در شکل ۴,۳۴ نشان داده شده است.



شکل ۴,۳۴. گراف جهت دار متناظر با مدل پتری ۴,۳۳

همانند قبل، برای نشان دادن خاصیت همبندی قوی در گراف جهت دار از الگوریتم کزاراجو استفاده می کنیم. در این روش با جستجوی اول عمق گراف، هر گره را پس از خاتمه مشاهده، در پشته ثبت می کنیم. دنباله حاصل از عملیات جستجوی اول عمق و نیز محتویات پشته پس از پایان عملیات به صورت زیر خواهند بود:

DFS= {0,1,2,3,4,5,6}

Stack= 6,5,4,3,2,1,0

سپس با ترانهاده کردن گراف، باز هم عملیات جستجوی اول عمق را با شروع از P0 بر روی آن اجرا می کنیم. با مشاهده هر گره، آن را از پشته و از گراف حذف کرده و آن را در یک گروه همبند قوی با حضور P0 قرار می دهیم.

در پایان می بینیم که همه گره ها در یک گروه همبند قوی با حضور P0 هستند. این بدان معنی است که گراف همبند قوی است.

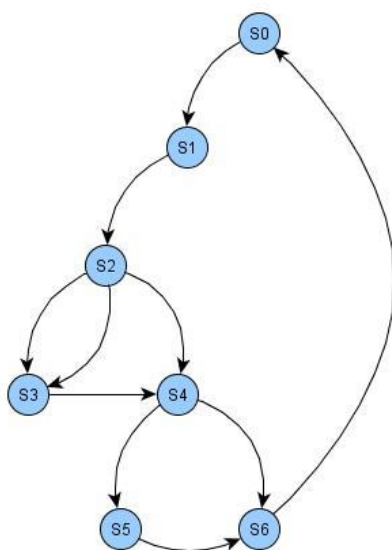
با توجه به نتیجه به دست آمده و نیز حضور حداقل یک توکن در وضعیت اولیه M_0 (در P0)، می توان به این نتیجه رسید که شبکه پتری ۴,۳۳ دارای خصوصیت Liveness است.

Safeness .۴,۹,۳,۲

با توجه به قضیه ۵، شرط لازم و کافی برای safe بودن یک live state machine، حضور دقیقاً یک توکن در شبکه در وضعیت M_0 است. این شرط نیز کاملاً برآورده شده است. بنابراین بدون نیاز به استدلال بیشتر، می توان از Safe بودن شبکه مطمئن بود.

Reversibility .۴,۹,۳,۳

برای بررسی این خصوصیت، باید از وجود خاصیت همبندی قوی در گراف پوشای مربوط به مدل پتری مطمئن شویم. این گراف پوشا در شکل ۴,۳۵ آمده است.



شکل ۴,۳۵. گراف پوشای مدل پتری ۴,۳۳

با اعمال الگوریتم کزاراجو، می بینیم که گراف همبند قوی است. بنابراین می توان نتیجه گرفت که reversible نیز می باشد.

با توجه به اینکه این مدل پتری دارای هر سه خاصیت مورد بررسی می باشد، این شبکه خوش رفتار است.

۴,۱۰. جمع بندی

در این فصل با تکیه بر مقدمات گفته شده در فصل ۳ درباره تکنولوژی های مجازی سازی، یک نمونه دیتا سنتر طراحی شد. برای ساده تر کردن تحلیل، این سیستم تا حد امکان کوچک اما در بر دارنده تمام اجزاء یک دیتا سنتر واقعی است. در مرحله بعد، با استفاده از اطلاعات و توصیفات ارائه

شده در فصل ۲ درباره ابزار مدل‌سازی فرمال، یک مدل پتری از کل دیتا سنتر تهیه شد. با توجه به اینکه دیتا سنتر یک سیستم موازی و دینامیک است، به کمک این مدل فرمال توانستیم خصوصیات رفتاری سیستم از جمله Liveness، Safeness و Reversibility سیستم را بررسی کنیم. در بخش‌های بعدی این فصل، به سراغ تک‌تک سرویس‌ها و مکانیزم‌های تشکیل‌دهنده ساختار مجازی‌سازی در دیتا سنتر مذکور رفته و مدل‌های پتری متعددی از آنها در چندین لایه تهیه نمودیم تا به این وسیله خصوصیات رفتاری آنها را نیز بررسی کنیم. در این مسیر، از قضایا و ابزارهای موثر بر روی شبکه‌های پتری و نیز در حالت کلی تر، از قضایای مربوط به گراف‌های جهت‌دار به دفعات استفاده شده است.

فصل پنجم: نتیجه گیری و پیشنهادات

در این پایان نامه، از زبان فرمال شبکه های پتری برای مدل سازی لایه مجازی سازی در دیتا سنترها استفاده شده است تا به کمک آن ها، خصوصیات رفتاری ماژول های این سیستم بررسی و مطالعه شود. برای این کار یک دیتا سنتر نمونه با خصوصیات دیتا سنترهای واقعی طراحی شده و سپس از مکانیزم ها و ماژول های مختلف نرم افزاری و مجازی سازی آن مدل های پتری تهیه شده است. بخش اول این ماژول ها شامل سرویس های سطح بالایی است که بر روی کل ماشین ها کار می کنند، از جمله HA، FT، DRS و غیره. بخش دوم شامل مکانیزم های مختلف Hypervisor، که بر روی هر یک از سرورهای فیزیکی نصب شده اند، شامل ساختار شبکه، ساختار ذخیره سازی و غیره می باشد.

بخش اعظم این تحقیق به بررسی رفتار و نحوه گردش کنترل در ماژول های سیستم مجازی سازی دیتا سنتر اختصاص دارد. به این ترتیب که این انتقال کنترل و تغییر وضعیت سیستم را از جنبه های مختلفی می توان بررسی نمود. اگر خصوصیات یک سیستم دینامیک و موازی مانند سرویس های مذکور را بدون در نظر گرفتن وضعیت شروع سیستم بررسی کنیم، به آن خصوصیات ساختاری گفته می شود. در مقابل، اگر خصوصیات سیستم را با توجه به وضعیت شروع آن تحلیل نمائیم، در حال بررسی خصوصیات رفتاری هستیم.

در این پایان نامه، پس از طراحی مدل پتری برای هر یک از سرویس ها و ماژول های سیستم، خصوصیات رفتاری آن بررسی و تحلیل شد. خصوصیات بررسی شده برای هر یک از بخش ها عبارتند از: Liveness، Safeness (و یا حداقل Boundedness) و Reversibility. در صورتی که

سیستم یا زیر سیستمی هر سه خصوصیت مذکور را داشته باشد، یک سیستم خوش رفتار است. این بدان معنی است که ماژول مذکور در حین کار هرگز دچار بن بست و سرریز نخواهد شد و نیز همیشه امکان بازگشت به نقطه شروع را خواهد داشت.

جدول ۵،۱ نتیجه حاصل از تحلیل رفتار ماژول های بررسی شده در فصل چهار را نشان می دهد.

جدول ۵،۱. خلاصه نتایج به دست آمده از تحلیل رفتار زیر سیستم ها و سرویس های VMware ESX

خوش رفتاری	خصوصیات رفتاری				نام ماژول	بخش مورد بررسی	ردیف
	Reversibility	Safeness	Boundedness	Liveness			
خیر	خیر	بله	بله	بله	مدل abstract از دیتا سنتر	۴،۲	۱.
خیر	خیر	بله	بله	خیر	سرویس High Availability	۴،۳	۲.
خیر	بله	خیر	بله (2-bounded)	بله (L3-live)	سرویس Fault Tolerance	۴،۴	۳.
بله	بله	بله	بله	بله	سرویس VMotion	۴،۵	۴.
بله	بله	بله	بله	بله	ESX Hypervisor	۴،۶	۵.
بله	بله	بله	بله	بله	سیستم ذخیره سازی در ESX	۴،۷	۶.
بله	بله	بله	بله	بله (L3-live)	ساختار شبکه در ESX	۴،۸	۷.
بله	بله	بله	بله	بله	سوئیچ مجازی (vSwitch)	۴،۹	۸.

برای تحقیق بیشتر در این حوزه می توان موضوعات بسیاری را در ادامه این تحقیق پیشنهاد نمود که به شناخت بیشتر لایه مجازی سازی در دیتا سنترها و تهیه مدل های فرمال دقیقتر از آنها کمک می کند. با در نظر گرفتن جنبه های بیشتری از سیستم می توان مدل های پتری دقیقتری از آن تهیه کرده و خصوصیات رفتاری و ساختاری بیشتری را تحلیل نمود. در ادامه به تشریح موارد قابل بررسی بر روی مدل های پتری تهیه شده از این سیستم می پردازیم.

۱- با وارد کردن پارامتر "زمان" در مدل سازی می توان نحوه توزیع توکن ها در مدل بر حسب گذشت زمان داخلی سیستم و نیز زمان خارجی را بررسی نمود. تحلیل سیستم از این دیدگاه به کمک کلاس خاصی از شبکه های پتری تحت عنوان شبکه های پتری زمان دار امکان پذیر است که در آن با نسبت دادن یک بازه زمانی به گزارها، نحوه توازی و همگامی پروسه های در حال اجرا در سیستم بررسی می شوند [۷۵] و [۷۶].

۲- با اضافه کردن احتمالات و وقایع آماری به نحوه اجرا شدن گزارهای مدل، کلاس دیگری از شبکه های پتری حاصل می شود که به آن شبکه های پتری آماری گفته می شود. به کمک این شبکه ها می توان رفتار غیر تعینی و احتمالی سیستم را بررسی نمود. این کار به بررسی رفتار سیستم در بلند مدت کمک می کند. از دیگر کاربردهای مهم این نحوه بررسی سیستم، سنجش میزان کارایی آن است [۷۷] و [۷۸].

۳- با بررسی دقیقتر invariantها می توان اطلاعات دقیقتری در خصوص نحوه کارکرد سیستم و مشکلات آن به دست آورد. با این توضیح که خصوصیات در سیستم که با تغییر وضعیت موقعیت ها در شبکه ثابت هستند، خصوصیات مستقل از موقعیت^۵ و خصوصیات مستقل از تغییر وضعیت گزارها در شبکه ثابت می مانند، خصوصیات مستقل از گزار نامیده می شوند. این ابزار که به کمک ماتریس مجاورت و معادلات متجانس قابل

¹ Timed Petri nets

² Stochastic Petri net

³ Non deterministic

⁴ Performance evaluation

⁵ Place invariants (P-invariant)

⁶ Transition invariants (T-invariant)

بررسی و تحلیل است، به شناخت خصوصیات ساختاری سیستم بسیار کمک می کند. این ابزار و نیز ابزار trap، در مواقعی که گراف پوشای یک مدل پتری بزرگ و پیچیده و حتی غیرقابل استخراج است، به تحلیل وجود بن بست و سرریز در سیستم بسیار کمک می کند [۲۳]، [۷۹] و [۸۰].

در پایان باید تاکید نمود که با وارد کردن هر چه بیشتر پارامترهای سیستم واقعی در فرایند مدل سازی، تصویر فرمال دقیقتری از سیستم به دست می آید. سپس با بررسی هر چه دقیقتر این مدل فرمال به استخراج دانش عمیقتری در مورد رفتار و چگونگی سیستم و بهینه نمودن آن در مراحل قبل از طراحی، در حین طراحی و پس از آن در طول حیات سیستم دست خواهیم یافت. در این راه، متدلوژی های مختلف تحلیل از جمله تجزیه مدل به بخش های ساده تر، طراحی چند لایه، انتزاع و نشان دادن حجم فراوانی از اطلاعات در نمایش کوچکی از مدل (مانند استفاده از ابزار مدلسازی سطح بالا- شبکه های پتری شئی گرا^۱ و مامور گرا^۲) و نیز ابزار ریاضی گفته شده کمک های شایانی خواهند کرد.

¹ Object-Petri nets

² Agent oriented Petri nets

- [1] W. Tschudi, T. Xu, D. Sartor, J. Stein “High-Performance Data centers, a research roadmap”, Lawrence Berkeley National Laboratory, University of California, LBNL-53483,
- [2] M. Bailey, “The Economics of Virtualization: Moving Toward an Application-Based Cost Model”, White Paper. MA, USA: IDC, November 2009.
- [3] C. Chaubal , “The Architecture of VMware ESXi”, White paper, VMware Inc., Revision: 20080527 WP-030-PRD-01-02 , 2007.
- [4] Jim Woodcock, Peter G. Larsen, Juan Bicarregui, John Fitzgerald, “Formal Methods: Practice and Experience”, ACM Computing Surveys (CSUR), Volume 41 Issue 4, October 2009.
- [5] John D. Gannon, James M. Purtilo, Marvin V. Zelkowitz, “SOFTWARE SPECIFICATION: A Comparison of Formal Methods”, Department of Computer Science, University of Maryland, Feb. 2001.
- [6] Mariagrazia Dotoli, Maria Pia Fanti, “An urban traffic network model via coloured timed Petri nets”, Elsevier, The Seventh Workshop On Discrete Event Systems (WODES2004), Volume 14, Issue 10, October 2006, Pages 1213–1229
- [7] A. Pouyan, H. T. Shandiz, S. Arastehfar, “Synthesis a Petri net based Control model for a FMS cell”, Elsevier, Computers in Industry, Volume 62, Issue 5, June 2011, Pages 501-508, Elsevier Science Publishers B. V. Amsterdam, Netherlands.
- [8] Examples of Industrial Use of CP-nets, http://www.daimi.au.dk/CPnets/intro/example_indu.html
- [9] Dong Seong Kim, Fumio Machida, Kishor S. Trivedi, “Availability Analysis of a Virtualized Data Center”, ICVCI 2009.
- [10] Ian Sommerville, “*Formal Specification*”, *Software Engineering, 7th edition*, School of Computer and Communication, university of Lancaster, 2004.
- [11] Eyal Oren, Armin Haller, “Formal frameworks for workflow modeling”, DERI Technical Report 2005-04-07, April 2005.
- [12] Egon Börger, "High Level System Design and Analysis using Abstract State Machines, Current Trends in Applied Formal Methods (FM-Trends 98). Springer LNCS 1641, 1999, pp. 1-43.
- [13] Yuri Gurevich. “Evolving Algebras: An Attempt to Discover Semantics”, EATCS 1 Bulletin, volume 43, pages 264-284, European Assoc. for Theor. Computer Science, Feb. 1991. (The Tutorial).
- [14] Uwe Glässer, Yuri Gurevich, Margus Veanes, “An Abstract Communication Model”, Microsoft Research Technical Report MSR-TR-2002-55, May 2002.

- [15] Kamran Sartip, "LOTOS: Language of Temporal Ordering Specification", Department of Computing and Software, McMaster University, Canada, March 2007.
- [16] Kenneth J. Turner, "*The Formal Specification Language LOTOS: A Course For Users*", Department of Computing Science and Mathematics, University of Stirling, Scotland, Aug. 1989.
- [17] L. Logrippo, M. Faci, M. Haj-Hussein, "*An introduction to LOTOS: learning by examples*", Computer Networks and ISDN Systems, Volume 23, Issue 5, Feb. 1992, pp 325-342, ISSN 0169-7552, DOI: 10.1016/0169-7552(92)90011-E.
- [18] Bernhard K. Aichernig, Andreas Kerschbaumer, "Property Orientation in the Model Oriented Vienna Development Method (VDM)", Siteceer X, 1999, [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.4913>
- [19] Marian Iordache, Panos Antsaklis, "A Survey on the Supervision of Petri Nets," DES Workshop PN 2005, Miami, FL, June 21, 2005.
- [20] Marian V. Iordache and Panos J. Antsaklis, "Petri Nets and Programming: A Survey", 2009 American Control Conference, Hyatt Regency Riverfront, St. Louis, MO, USA, June 10-12, 2009.
- [21] T. Miyamoto, S. Kumagai, "A Survey of Object-Oriented Petri Nets and Analysis Methods", IEICE TRANS. FUNDAMENTALS, VOL. E88-A, NO.11 NOVEMBER 2005.
- [22] R. Alur, J. Esposito, M. Kim, V. Kuma, I. Lee, "Formal modeling and analysis of hybrid systems: A case study in multi-robot coordination", FM'99: Proceedings of the World Congress on Formal Methods, LNCS 1708, pp. 212--232, Springer, 1999.
- [23] T. Murata, "Petri Nets: Properties, Analysis, and Applications", Proceedings of IEEE, 77 (4) (1989) 541--579.
- [24] R. Zurawski, M. Zhou, "Petri Nets and Industrial Applications: A Tutorial", IEEE. Trans. Industrial Electronics, Vol. 41, No. 6, Dec. 1994.
- [25] W.M.P. van der Aalst, K.M. van Hee, G.J. Houben, "Modelling and Analysing Workflow using a Petri-net based approach", Proceedings of Second Workshop on Computer-supported Cooperative Work, Petri nets related formalisms, pp. 31-50, 1994.
- [26] Emine G. Aydal, Richard F. Paige, Mark Utting, Jim Woodcock, "Putting Formal Specifications under the Magnifying Glass: Model-based Testing for Validation", IEEE Proc. ICST '09 on Software Testing Verification and Validation, 2009, ISBN: 978-0-7695-3601-9
- [27] C. Girault, R. Valk, "Petri Nets for Systems Engineering, A Guide to Modelling, Verification, and Applications", ISBN: 3540412174, Springer-Verlag, July 2001.

- [28] L. E. Holloway, B. H. Krogh and A. Giua, "A Survey of Petri Net Methods for Controlled Discrete Event Systems", SpringerLink journal on Discrete Event Dynamic Systems, Vol. 7, Issue 2, April 1997, pp. 151-190, DOI: 10.1023/A:1008271916548.
- [29] Yuri Gurevich. *Evolving Algebras*. In B. Pehrson and I. Simon, editors, *IFIP 1994 35 World Computer Congress, volume I: Technology and Foundations*, pages 423-427. Elsevier, Amsterdam, 1994.
- [30] E. B"orger, "Modeling with Abstract State Machines: A support for accurate system design and analysis", Modellierung 2004, volume P-45 of GI-Edition Lecture Notes in Informatics, pages 235–239. Springer-Verlag, 2004.
- [31] E. B"orger, S. Mazzanti, "A practical method for rigorously controllable hardware design", ZUM'97: The Z Formal Specification Notation, volume 1212 of Lecture Notes in Computer Science, pages 151–187. Springer-Verlag, 1997.
- [32] David Lee, Mihalis Yannakakis, "Principles and methods of testing finite state machines-a survey", IEEE Proc. Vol. 84, Issue: 8, pp. 1090 – 1123, ISSN: 0018-9219, Aug. 1996.
- [33] Tommaso Bolognesi, Ed Brinksma, "Introduction to the ISO specification language LOTOS", ACM Journal on Computer Networks and ISDN Systems, Volume 14, Issue 1, March, 1987, doi: 10.1016/0169-7552(87)90085-7.
- [34] Guy Ledu, "Multimedia in the E-LOTOS Process Algebra", Chapter 16 of "Formal Methods For Distributed Processing - A Survey of Object Oriented Approaches", Cambridge University Press, 2001, pp. 357-372
- [35] Peter Gorm Larsen, Kenneth Lausdahl, Nick Battle, John Fitzgerald, "VDM-10 Language Manual", Overture Technical Report Series, No. TR-001, May 2010.
- [36] John Fitzgerald, Peter Gorm Larsen and Shin Sahara, "Technical report series: Modeling and Analysis in VDM", Proceedings of the Fourth VDM/Overture Workshop, University of Newcastle upon Tyne, No. CS-TR-1099, May 2008.
- [37] Massimo, "A brief architecture overview of VMware ESX, XEN and MS Viridian", [Online]. Available: <http://it20.info/2007/06/a-brief-architecture-overview-of-vmware-esx-xen-and-ms-viridian>.
- [38] VMeare, "A Performance Comparison of Hypervisors", VMware Inc., Performance study, Revision: 20070201 Item: PS-004-INF-01-002, 2007.
- [39] Tim Abels, Puneet Dhawan, Balasubramanian Chandrasekaran, "An Overview of Xen Virtualization", Dell Power Solutions, Aug. 2005.
- [40] Microsoft, "Hyper-V Architecture", Microsoft MSDN, [Online]. Available: <http://msdn.microsoft.com/en-us/library/cc768520%28v=bts.10%29.aspx> , 2011
- [41] Microsoft Inc., CA Inc., "Protecting your Virtualized Server Investment ", White paper, May 2010.

- [42] Microsoft Inc., “About Virtual Machines and Guest Operating Systems”, Microsoft MSDN, Sep. 2011, [Online]. Available: <http://technet.microsoft.com/enus/library/cc794868%28WS.10%29.aspx>.
- [43] Yuval Shavit, “Advantages and disadvantages of Hyper-V”, Search Systems Channel, [Online]. Available: <http://searchsystemschannel.techtarget.com/feature/Advantages-and-disadvantages-of-Hyper-V>
- [44] Mike Neil, “Hypervisor, Virtualization Stack, And Device Virtualization Architectures”, Windows Virtualization Microsoft Corporation, Microsoft WinHEC 2006.
- [45] CA Inc., “Deployment Options for Microsoft Hyper-V Server”, CA ARCSERVE REPLICATION & CA ARCSERVE HIGH AVAILABILITY R16, 2011.
- [46] CA Inc., “CA ARCserve Replication and High Availability Deployment Options for Microsoft Hyper-V”, CA ARCserve Replication and CA ARCserve High Availability, 2010.
- [47] Ray Weinstein, Burk Buechler, “Highly Available virtualization With Microsoft Hyper-v And scvMM 2008”, Dell Power Solutions, Nov. 2008.
- [48] Robert Larson, Janique Carbone et. All, “Windows Server 2008 Hyper-V Resource Kit”, Microsoft Press, ISBN: 0-7356-2517-4, June 2009.
- [49] Ian Pratt, “Xen and the Art of Virtualization”, University of Cambridge and Founder of XenSource Inc., 2006.
- [50] Paul Barham et. All, “Xen and the Art of Virtualization”, SOSP'03, October 19-22, 2003, Bolton Landing, New York, USA, Copyright 2003 ACM 1-58113-757-5/03/0010.
- [51] XenSource Inc., “Xen Architecture Overview”, White paper, [Online]. Available: <http://www.pdfio.com/k-461078.html>
- [52] Yaozu Dong et. All, “Extending Xen with Intel Virtualization Technology”, Intel Technology Journal, Vol. 10, Issue 3, ISSN: 1535-864X, Aug. 2006.
- [53] Ian Pratt, “Xen Status Report”, Description and status report on the Xen 3.0 development, University of Cambridge, April 2005.
- [54] Fengzhe Zhang, Jin Chen, Binyu Zang, “CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization”, Proceedings of Symposium on Operating Systems Principles (SOSP-2011), Cascais, Portugal, October 2011.
- [55] Diwaker Gupta, Ludmila Cherkasova, Rob Gardner, Amin Vahdat, “Enforcing Performance Isolation Across Virtual Machines in Xen”, Proceedings of the 7th ACM/IFIP/USENIX Middleware Conference. Melbourne, Australia, Nov. 2006.
- [56] Zhikui Wang, Xiaoyun Zhu, Pradeep Padala, Sharad Singhal, “Capacity and Performance Overhead in Dynamic Resource Allocation to Virtual Containers”, In

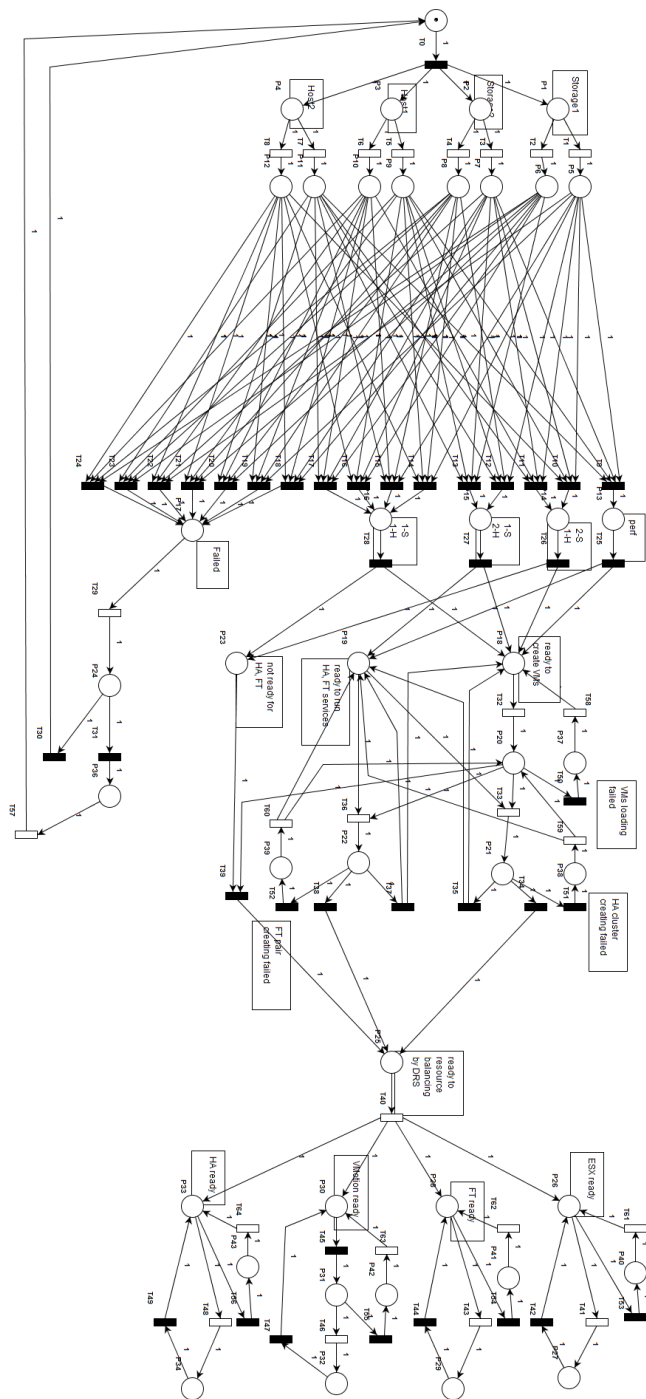
the proceedings of the IFIP/IEEE Symposium on Integrated Management (IM'2007).

- [57] Pradeep Padala, Xiaoyun Zhu, et al., “Automated Control of Multiple Virtualized Resources”, In the proceedings of the EuroSys 2009.
- [58] Timothy Wood, Prashant Shenoy, Arun Venkataramani, Mazin Yousif, “Black-box and Gray-box Strategies for Virtual Machine Migration”, Proceedings of the Fourth Symposium on Networked Systems Design and Implementation (NSDI), Cambridge, MA, April 2007.
- [59] VMware, “VMware Infrastructure Architecture Overview”, White paper, VMware, Inc., 2006
- [60] VMware, “Introduction to VMware vSphere”, VMware Inc., White paper, EN-000102-00, 2009.
- [61] VMware, “vSphere Availability Guide”, VMware Inc., White paper, EN-000108-03, 2009-2011.
- [62] VMware, “VMware Virtual Networking Concepts”, Information guide, VMware Inc., Revision: 20070718 Item: IN-018-INF-01-01, 2007.
- [63] VMware, “SAN System Design and Deployment Guide, third Edition”, Cookbook, VMware Inc., May 2010.
- [64] Nikhil Bhatia et al., “VMware vCenter Server Performance and Best Practices”, Performance study, EN-000435-00, VMware Inc., 2010.
- [65] VMware, “Best Practices for VMware ESX Server 3”, White paper, VMware Inc., Revision: 20060605 Item: ESX-ENG-Q206-236, 1998.
- [66] Iwan Rahabok, “Private Cloud, Sample Architectures”, VMware Inc., Singapore, Mar. 2011.
- [67] Warren Ponder, Fred Schimscheimer, John Dodge, “VMware View, A Guide to Large-scale Enterprise VMware View 3 and VMware View 4 Deployments”, Reference architecture, VMware Inc., Revision: 20070719 Item: TBD-D1, 2010.
- [68] VMware, “Architecture Brief: The Brocade Data Center Fabric Reference Architecture for VMware vSphere 4”, Architecture brief, Brocade Inc., 2009.
- [69] VMware, “VMware ESX and VMware ESXi”, Product datasheet, VMware Inc., Item No: VMW_09Q3_DS_ESX_ESXi_USLET_EN_P6_R4, 2009.
- [70] VMware, “VMware VMotion and CPU Compatibility”, Information guide, VMware Inc., Revision 20080609 Item: EN-000039-00, 2008.
- [71] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, “Introduction to Algorithms”, MIT press, ISBN: 978-0-262-03384-8, 1990.
- [72] VMware, “Virtualization Infrastructure”, Tutorial, VMware Inc.,
- [73] Cisco, VMware, “VMware Infrastructure 3 in a Cisco Network Environment”, DCisco Systems, Inc., OL-16503-01, 2008.

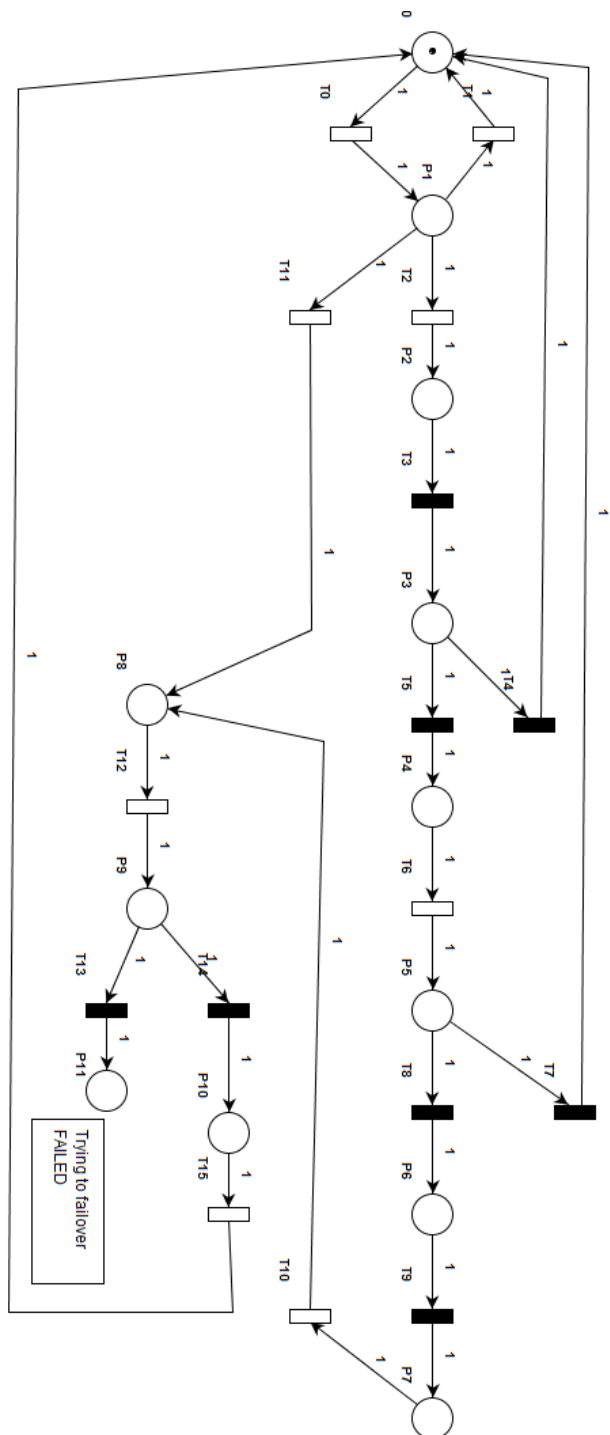
- [74] Sreekanth Setty, “VMware Network I/O Control: Architecture, Performance and Best Practices”, Technical white paper, VMware Inc., Item No: VMW_10Q3_WP_NETIOC_BESTPRACTICES_p34__A_R2, 2010.
- [75] Miryam Barad, “TIMED PETRI NETS AS A VERIFICATION TOOL”, IEEE WSC '98 Proceedings of the 30th conference on Winter simulation, ISBN:0-7803-5134-7, 1998.
- [76] Patricia Bouyer, Serge Haddad, Pierre-Alain Reynier, “Timed Petri Nets and Timed Automata: On the Discriminating Power of Zeno Sequences”, Information and Computation, 206(1):73—107, 2008.
- [77] Peter J. Haas , “Stochastic Petri Nets: Modelling, Stability, Simulation”, Springer-Verlag, ISBN 0-387-95445-7, 2002.
- [78] M. Ajmone Marsan, “STOCHASTIC PETRI NETS: AN ELEMENTARY INTRODUCTION”, SpringerLink, Advances in Petri Nets, Lecture Notes in Computer Science, Volume 424/1990, 1-29, DOI: 10.1007/3-540-52494-0_23, 1989
- [79] Fritz Krückeberg, Michael Jaxy, “Mathematical methods for calculating invariants in Petri nets”, SpringerLink, Advances in Petri Nets, Lecture Notes in Computer Science, Volume 266/1987, 104-131, DOI: 10.1007/3-540-18086-9_22, 1987.
- [80] J. Sifakis, “Use of Petri nets for Performance Evaluation”, 3rd Intl. Symposium on Modeling and Evaluation, IFIP, North Holland, pp. 75-93, 1977.

پیوست: مجموعه کامل مدل های پتری طراحی شده در پایان نامه

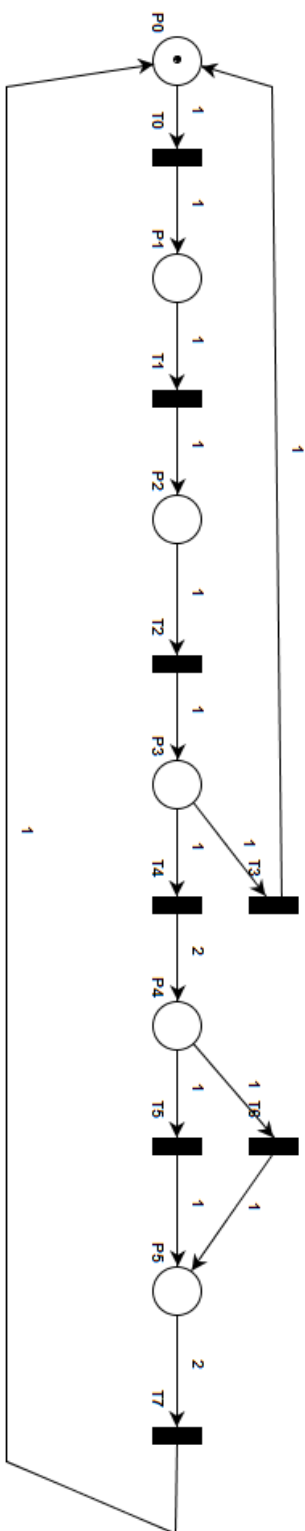
۱. مدل پتری طراحی شده برای دیتا سنتر نمونه



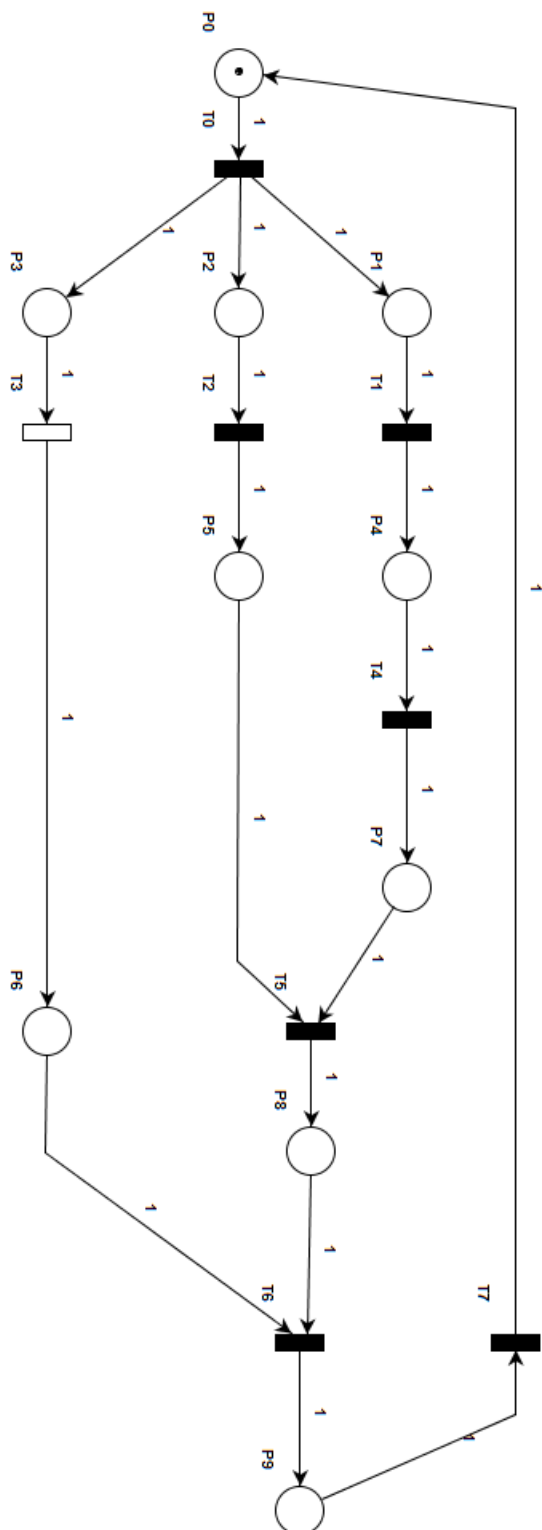
۲. مدل پتری نحوه کار سرویس HA



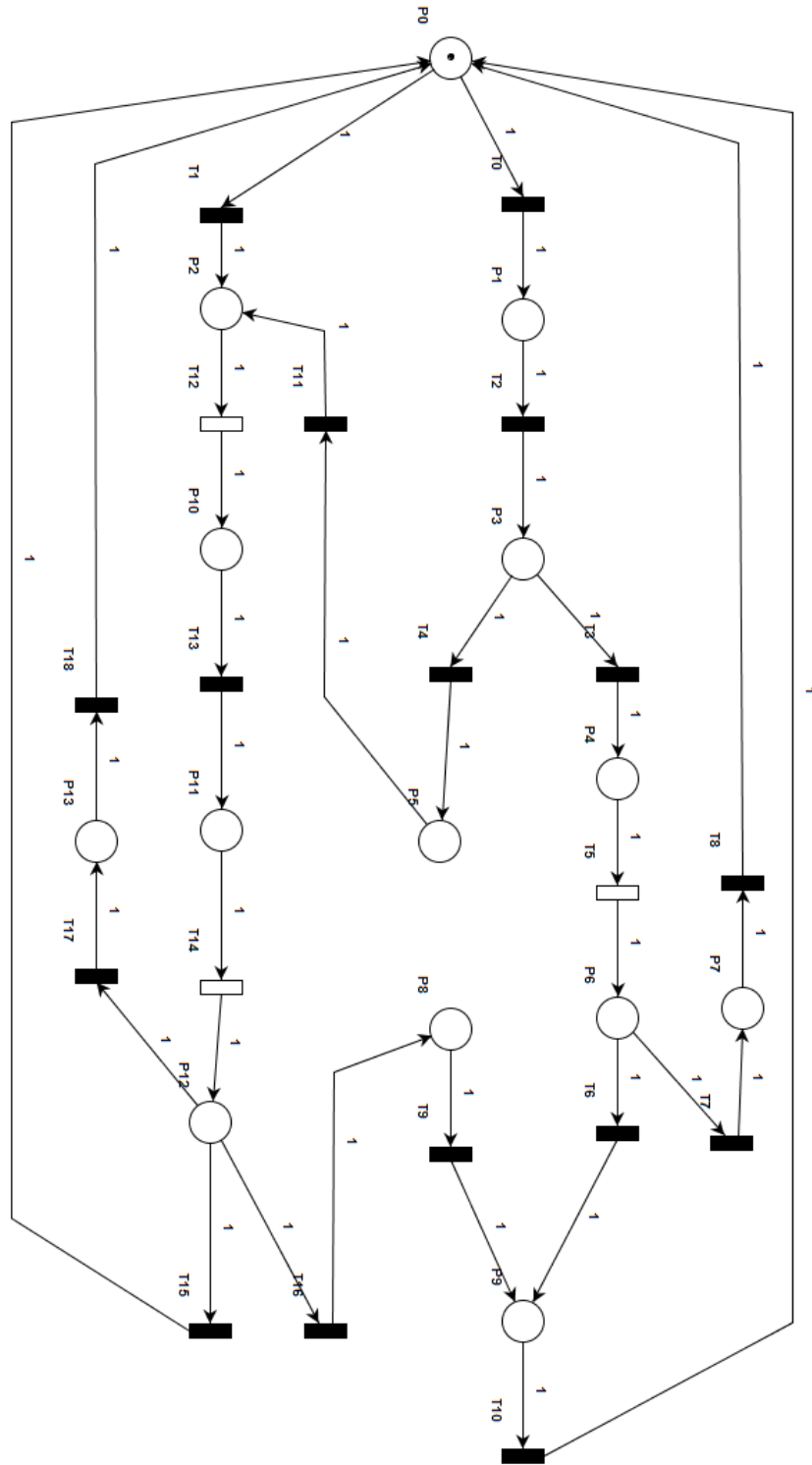
۳. مدل پتری نحوه کار سرویس Fault Tolerance



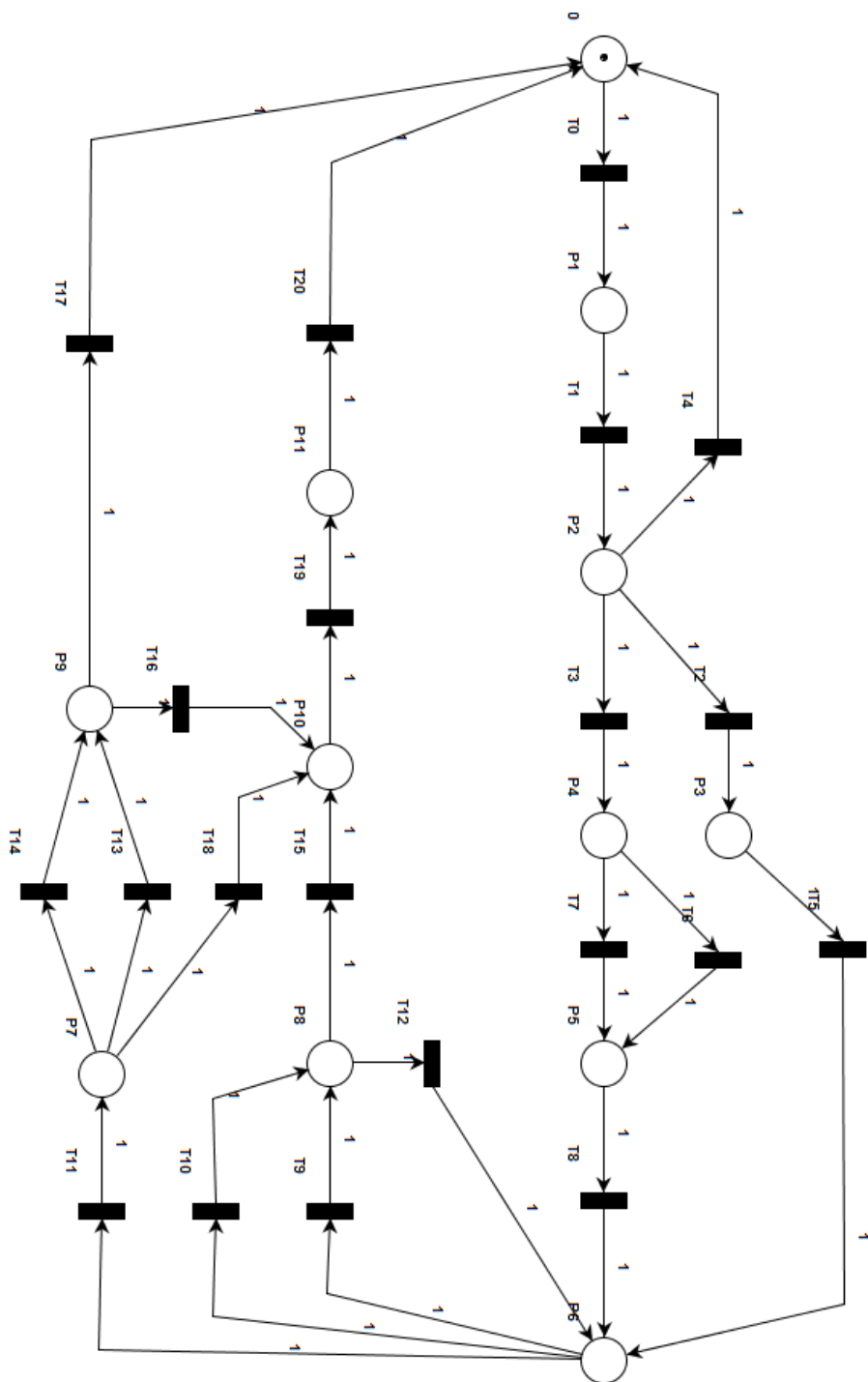
۴. مدل پتری نحوه کار سرویس VMotion



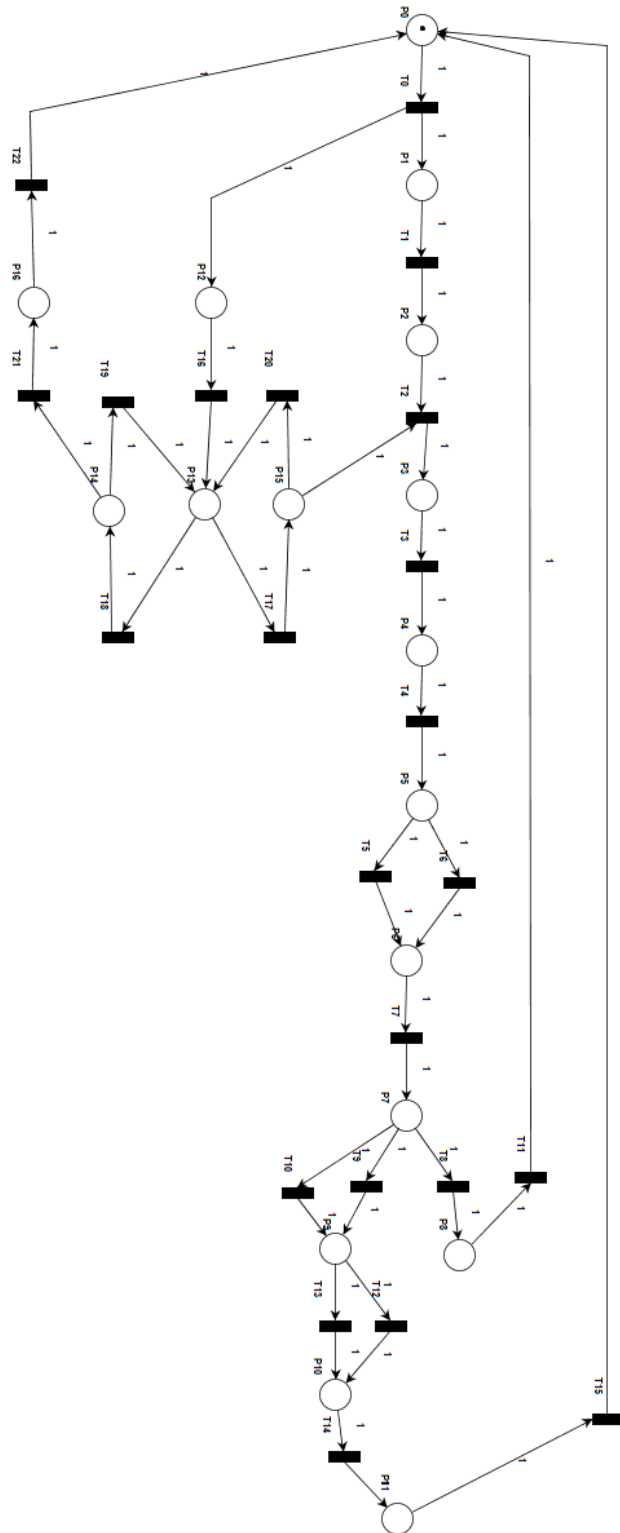
۵. مدل پتری طرز کار ESX



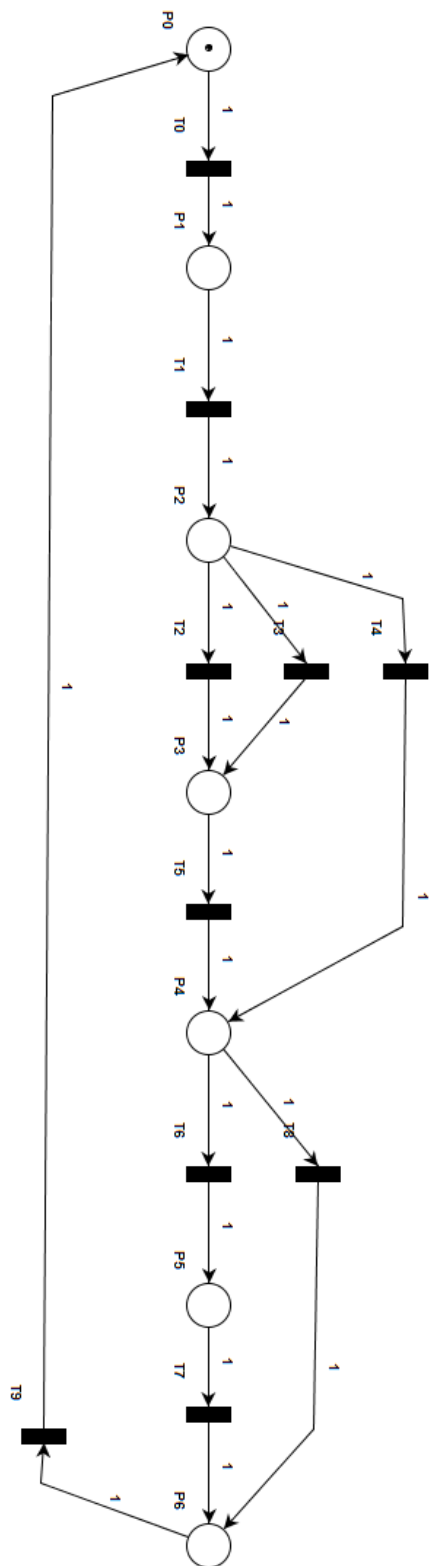
۶. مدل پتری ارائه شده از نحوه کار سیستم ذخیره سازی در ESX



۷. مدل پتری تهیه شده از ساختار شبکه در ESX



۸. مدل فرمال از نحوه کار سوئیچ مجازی



Abstract

A Formal Model for Analysis of the ICT Infrastructure in Data Centers

By:

Ali HassanBeigi

In this research, formal models for virtualization layer in data centers have been developed. These models have been aimed to provide means and analytical tools for studying and analyze the components and services in the virtualization layer, more precisely and formally. By means of Petri nets formal specification language, behavioral properties of each layer can be studied and analyzed, as well. Studying properties like Liveness, Safeness and Reversibility in the formal models constructed for real world scenarios of each module in virtualization structure, led to extracting a large volume of knowledge. The generated knowledge covers many aspects of the real world phenomena. It captures system behavior, bottlenecks, overflows, synchronization status, performance measure and other structural and behavioral aspects of the system under study. The focus in this research is mainly on the modeling and analysis of the virtualization layer in typical data centers to capture the above aspects of the system.

IN THE NAME OF GOD

**A Formal Model for Analysis of the ICT Infrastructure in Data
Centers**

BY
ALI HASSANBEIGI

THESIS
SUBMITTED TO THE SCHOOL OF GRADUATE STUDENTS IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER
OF SCIENCE (M. SC.)

IN
INFORMATION TECHNOLOGY ENGINEERING
(E-COMMERCE)

SHIRAZ UNIVERSITY

SHIRAZ

ISLAMIC REPUBLIC OF IRAN

EVALUATED AND APPROVED BY THE THESIS COMMITTEE AS:

..... A. POUYAN, Ph. D., ASSISTANT PROF. OF COMPUTER
ENGINEERING & IT, SHAHROOD UNIVERSITY OF
ECHNOLOGY (CHAIRMAN)

.....F. KHUNJUSH, Ph. D., ASSISTANT PROF. OF
COMPUTER ENGINEERING & IT, SHIRAZ UNIVERSITY

.....M. KESHTGARI, Ph. D., ASSISTANT PROF. OF COMPUTER
ENGINEERING

January 2012



Shiraz University
Faculty of e-Learning

M.S.Thesis in Information Technology
Engineering (e-Commerce)

A Formal Model for Analysis of the ICT
Infrastructure in Data Centers

By
Ali HassanBeigi

Supervised by
Ali A. Pouyan

January 2012