حافظه مشترک توزیع شده مبتنی بر مهاجرت زنده ماشین مجازی

چکیده

مهاجرت زنده ماشین مجازی یک ابزار حیاتی برای مدیریت پویای منابع در مراکز داده فعلی است. تکنیک های بسیاری برای رسیدن به این هدف با حداقل وقفه در امر سرویس دهی توسعه یافته اند. در این مقاله یک پیش نویس از مهاجرت زنده VM با استفاده از مدل محاسباتی حافظه مشترک توزیع شده (DSM) ارائه میشود. که به وسیله دو گره محاسباتی یکسان برای ایجاد معماری خدمات محیطی یعنی زیرساخت های مجازی سازی، سرور ذخیره سازی مشترک و DSM و خوشه (HPC) با کارایی محاسباتی بالا راه اندازی میگردد. چارچوب سفارشی DSM براساس یک به روز رسانی حافظه Grappa با زمان تاخیر کم می باشد. خوشه HPC با کتابخانه های OPENMPI و MPI از موازی سازی و موازی سازی خودکار حجم کار با استفاده از گره های محاسباتی پردازنده ها پشتیبانی میکند.DSM به پردازنده های خوشه اجازه می دهد تا به صفحات مشابه حافظه دسترسی داشته و در نتیجه به روز رسانی داده حافظه بر اساس به روز رسانی های ویژگی های محلیت کمتر باشد، که باعث کاهش حجم داده منتقل شده در شبکه می گردد. این مدل بهبود خوبی در معیارهای مهاجرت زنده VM بدست آورده است. زمان خرابی در زمان بیکاری ویندوز VM ٪50 و در زمان بیکاری لینوکس اوبونتو 66.6٪ کاهش یافته است. به طور کلی، این مدل نه تنها باعث کاهش مدت زمان خرابی و مقدار کل داده های ارسال شده میشود، بلکه معیارهای دیگری مانند مجموع زمان مهاجرت و کارایی نرم افزار را کاهش نمیدهد.

کلمات کلیدی: ماشین مجازی، مجموع مدت زمان مهاجرت، زمان خرابی، حافظه مشترک توزیع شده، ماشین فیزیکی، HPC.

1. مقدمه و کارهای پیشین

روند مهاجرت زنده ماشین مجازی (VM) از سرویسهای عمده ارائه شده توسط ارائه دهندگان خدمات ابری مدرن است. می توان آن را به عنوان انتقال وضعیت ماشین مجازی (VM) تعریف نمود، در حالی که همچنان در حال اجرا و سرویس دهی به مشتریان از یک ماشین فیزیکی به دستگاه فیزیکی دیگر بدون هیچگونه اختلال دسترسی می باشد. وضعیت VM به طور پویا در طول پروسه مهاجرت زنده تغییر میکند. در نتیجه خدمت رسانی زنده به مشتریان، این تغییرات وضعیت حافظه، رجیسترها و وضعیت پردازنده مجازی VM (vCPU) و وضعیت شبکه را تحت تاثیر قرار میدهد. برای انتقال امن این سه فضای کاری در حالی که همچنان VM در حال اجراست، باید این تغییرات را به یک شکل منسجم تا زمانی که شرایط توقف رخ دهد، ارسال نمود.

پژوهش های بسیاری از سال 2005 انجام شده است تا زمانی که Clark [1] روش پیش نسخه ای از مهاجرت زنده را پیشنهاد داد، که براساس تبادل مکرر حالت حافظه می باشد. پس از آن، روش های بهینه سازی زیادی[2] به منظور ارتقاء نحوه انتقال تصویر حافظه و وضعیت CPU ارائه شدند. رویکرد دیگر که مبتنی بر لاگ پردازنده وپاسخدهی است [3] نیاز به هماهنگ سازی داشته و کارهای زیادی با استفاده از این روش صورت نگرفته است. تمام این روشها چهار معیار عملکردی زیر را دربرداشتند:

1) مجموع زمان مهاجرت: مدت زمان شروع فرآیند مهاجرت تا پایان آن. هدف آن کاهش زمان کل مهاجرت است.

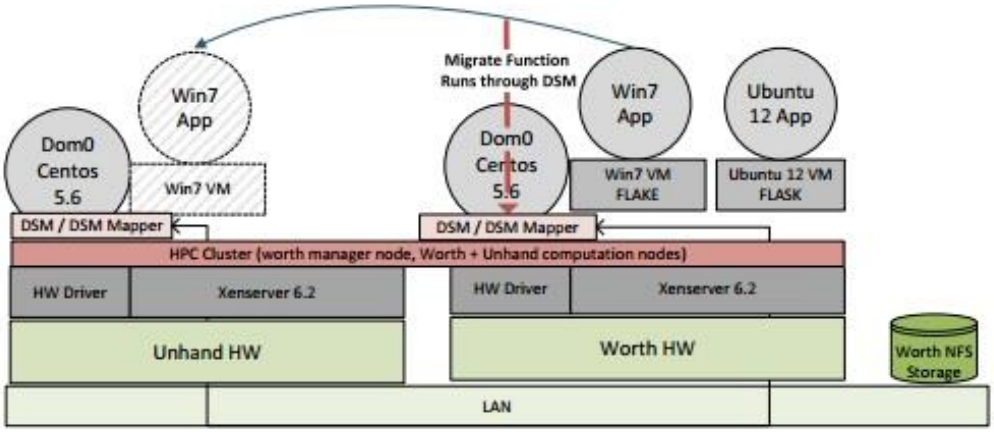
2) زمان خرابی: زمانی که اجرای vCPU در منبع دستگاه فیزیکی (PM) به حالت تعلیق درآمده تا وقتی که در مقصد PM از سر گرفته شود. هدف آن کاهش زمان خرابی می باشد.

3) حجم اطلاعات منتقل شده: حجم اطلاعاتی که در مدت زمان مهاجرت زنده در طول شبکه منتقل میشود. هدف آن کاهش حجم داده های منتقل شده است.

4) عملکرد کاربردی ماشین های مجازی: پاسخ نرم افزار VM مهاجرت کرده. هدف آن حفظ عملکرد نرم افزار می باشد. بخش های بعدی مدل ارائه شده و نحوه کار آن را توضیح میدهد.

2. ماژول های طراحی سیستم

در این مقاله، مهاجرت پیش نسخه ای از هایپروایزر Citrix Xen با استفاده از اجرای DSM در محاسبات با کارایی بالا (HPC) اعمال شده است. DSM مورد استفاده به تناسب نیازها اصلاح گردیده است. این معماری سیستمی چهار سرویس (NFS، هایپروایزر XenMotion، HPC و DSM) را اجرا میکند، که به شیوه ای همکارانه برای مهاجرت زنده VM از هایپروایزر XenServer در یک راه بهینه ، بهبود حرکت XenServer با استفاده از ذخیره سازی مشترک NFS ، و خوشه DSM HPC برای سرعت بخشیدن به حرکت XenServer VM عمل میکنند. بلوک های ساختار معماری پیشنهادی از سه لایه سرویس تشکیل شده که امر مهاجرت را از طریق اجرای روند مهاجرت مجازی به عنوان یک وظیفه در خوشه محاسباتی DSM HPC تسهیل می سازد. شکل 1 معماری مفهومی از اجزای بلوک ماژول و جریان ارتباطات را نشان می دهد.



شکل 1. مدل طراحی سیستم

الف. اجزای فیزیکی راه اندازی سیستم

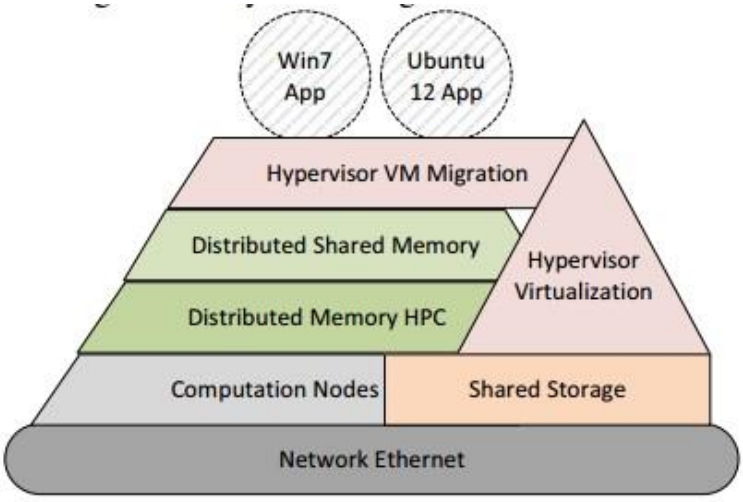
در این مقاله، دو ایستگاه کاری یکسان Dell با پردازنده سرعت بالا 4 هسته ای Xeon اینتل و سرعت 3.6 گیگاهرتز، که توسط سوئیچ اترنت Linksys با سرعت پورت Mbps100 استفاده شده اند. جدول 1 خلاصه ای از مشخصات سخت افزاری را نشان می دهد.

Dell Precision T1700 Specifications	
CPU	Intel(R) Xeon(R) CPU E3-1271 v3 @ 3.60GHz (4 Cores)
Memory (RAM)	32G Byte Kingston 1600 MHz (0.6 ns)
Storage (Hard Disk)	1T Byte ATA Disk
Network	Intel Ethernet Connection I217-LM 10/100/1000Mbps
OS	Citrix Xenserver (Linux Centos 5.6 Custom)
Xen Kernel	2.6.32.43-0.4.1.xs1.8.0.835.170778xen

جدول 1. مشخصات سخت افزاری

ب. بررسی منطقی

شکل 2 ماژول های منطقی را همانند لایه های معماری برای مهاجرت VM به تصویر می کشد. بخش اول پروتکل حافظه مشترک NFS است، که یک پروتکل شفاف بوده که اجازه می دهد تا به روز رسانی سرور ذخیره سازی مشترک با تمام اعضای مجازی هماهنگ گردد. بخش دوم زیرساخت مجازی سازی با استفاده از سرور Citrix Xen نسخه 6.2 است که برای ایجاد ماشین های مجازی مورد استفاده قرار گرفته و توسط Citrix Xen در مرکز کنسول مدیریت اداره میشود، که یک نرم افزار برای مدیریت ماشین های مجازی و قالب های مجازی می باشد. ذخیره سازی مشترک و ماژول های مجازی سازی راه اندازی اولیه برای مهاجرت زنده را فراهم می کنند.



شکل 2. ماژول های منطقی سیستم

بخش سوم، خوشه حافظه توزیع شده HPC به همراه تکنیک انتقال پیام است. به طور کلی در خوشه بندی HPC ، مفهوم حافظه توزیع شده برای پشتیبانی از برنامه نویسی موازی اجباری است، زیرا به استفاده از انتقال پیام (MP) برای ارتباط میان پردازنده ها نیاز دارد. ارتباط استاندارد میان پردازنده ها، رابط انتقال پیام (MPI) است که توسط تمام ارائه دهندگان محاسبات با کارایی بالا پشتیبانی می شود. نقش خوشه HPC ، ارائه موازی سازی و خدمات موازی سازی خودکار برای بخشهای کد، بر اساس ویژگی های کد با استفاده از کتابخانه OPENMPI است.

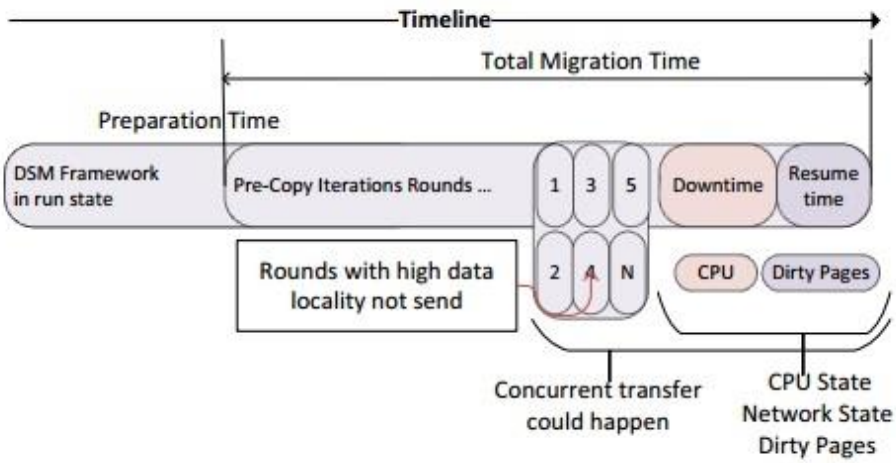
بخش چهارم چارچوب حافظه مشترک توزیع شده است. مدل DSM با خوشه HPC جهت ارائه دسترسی به حافظه مشترک برای تمام گره های خوشه و پردازنده ها عمل می کند. DSM ارتباطات بین پردازنده را بر اساس پارادایم ارتباطی حافظه مشترک تغییر میدهد، که اجازه می دهد تا تمام پردازنده ها در خوشه HPC به فضای حافظه مشابه دسترسی داشته باشند. در چنین الگویی ، مهاجرت فرآیند تنها به انتقال وضعیت فرایند از صف زمانبندی CPU در یک گره محاسباتی به صف آماده در گره پردازنده دیگر نیاز دارد، از آنجایی که بلوک کنترل فرآیند یا PCB ، کد و پشته در فضای آدرس حافظه مشابه قرار دارند و حافظه مجازی مشترک یک آدرس مستقل به اشتراک گذاشته شده توسط تعدادی از پردازنده ها می باشد. هر پردازنده می تواند به طور مستقیم به هر محل از حافظه در فضای آدرس مشترک دسترسی داشته باشد. نقش ماژول DSM ، ارائه وضعیت به روز رسانی حافظه دریک راه سازگار و منسجم است، که در آن روش پیش نسخه با انتقال مکرر صفحات حافظه VM شروع می گردد. DSM با اجتناب از ارسال صفحات خراب ، به صفحات در حال حرکت حافظه کمک می کند.

3. مهاجرت زنده VM با استفاده از DSM

دستگاه منبع (Worth) دو ماشین مجازی (FlakeوFlask) را اجرا می کند؛ سیستم عامل مهمان (Win7) از Flake مهاجرت خواهد کرد. این فرایند با تحریک تابع مهاجرت از XenServer XAPI آغاز میگردد که پس از آن فرایند مهاجرت برای اجرای پیش نسخه ای مهاجرت VM شروع خواهد شد. DSM دسترسی به حافظه مشترک را برای تمام گره های محاسباتی خوشه فراهم میکند که قادر به دسترسی مستقیم به تمام صفحات حافظه برای سرورهای

مجازی منبع و مقصد باشند. فرایند مهاجرت زنده VM توسط ماشین فیزیکی منبع متمرکز میشود، OPENMPI کد سریال را برای یافتن بخش های موازی به منظور افزایش سرعت اسکن میکند. DSM به عنوان یک فضای حافظه سراسری بین گره های محاسباتی خوشه HPC مورد استفاده قرار میگیرد که به فرآیند pulling مهاجرت VM کمک خواهد کرد. در همین حال دستگاه فیزیکی منبع همچنان به واگذاری وضعیت پردازنده VM مجازی و وضعیت شبکه با اولویت بالا ادامه داده و پس از آن انتقال صفحات خراب با اولویت پایین تر انجام خواهد شد. سرورهای مجازی سازی مبدا و مقصد با وظیفه مهاجرت برای شروع و خاتمه مهاجرت زنده VM بر اساس آستانه ثبات تصویر حافظه مقصد VM بارگزاری میشوند که توسط سه پارامتر محدود شده است: حداکثر تعداد تکرارها، تغییرات کمتر صفحات حافظه (تولید صفحات خراب کمتر)، و تصویر حافظه سازگار در مقصد.

DSM روند پیش نسخه مهاجرت زنده VM را با سفارشی سازی برخی از ویژگی های Grappa DSM بهبود می بخشد [4]. اولین تغییر ، به روز رسانی اندازه پیام هاست. فرایند مهاجرت DSM بر روی انتقال صفحات حافظه با متوسط اندازه 4کیلوبایت به جای 32 بایت عمل میکند همانطور که در Grappa استفاده می شود. به روز رسانی اندازه پیام همانند اندازه صفحات حافظه اجرا میشود. این امر تعداد پیام های به روز رسانی را کاهش می دهد. تغییر دوم به انتخاب صفحات حافظه بر اساس مدیریت محلی مربوط می شود. داده مورد نظر با محلیت پایین معمولا روی پردازنده خانگی خود اصلاح میگردد، به جای آنکه یک کپی از تصویر کد به پردازنده مقصد ارسال کند، که در آن داده مورد نظر با محلیت بالا به پردازنده درخواست شده ارسال میگردد. در روش پیشنهادی بهترین کار، انتقال صفحات حافظه با لوکالیتی بالا نیست، بلکه ارسال صفحات حافظه با لوکالیتی پایین با نرخ کمتر صفحات خراب به پردازنده مقصد، کارآمدتر می باشد. این مساله فرکانس به روز رسانی را کمتر خواهد نمود. صفحات حافظه خراب باقی مانده بالاتر، بعدا در حالت تعلیق و ادامه منتقل میشوند. موازی سازی خودکار ، ارسال و دریافت صفحات حافظه VM را بصورت موازی فراهم می کند. شکل 3 روش پیش نسخهDSM سفارشی را به صورت نمودار فضا زمانی توصیف کرده است. به این ترتیب رویکرد پیشنهادی تعداد صفحات خراب ارسالی از پردازنده خانگی به پردازنده مقصد را کاهش می دهد.



شکل 3. روند زمانی پیش نسخهDSM

4. معیارهای عملکرد

مهاجرت زنده VM برای هر دو سیستم عامل مهمان یعنی ماشین مجازی Flake Windows7 و Ubuntu12 لینوکس Flask ، بین دو سرور میزبان Unhand و Worth انجام گرفته است. این آزمایش برای هر مورد متفاوت ، شش بار اجرا شده تا مطمئن شویم که مقادیر اندازه گیری شده عاری از هر گونه خطاست، و سپس میانگین آن محاسبه شده است.

الف. معیارهای حجم کار VM

سیستم عامل مهمان VM با چهار حجم کاری مختلف برای تست مهاجرت زنده VM تحت سناریوهای مختلف بارگزاری شده، که مهاجرت VM را توسط برنامه های کاربردی که دارای رفتار اجرایی مختلفی هستند لود میکند، همانطور که در جدول 2 داده شده است.

Workload	Benchmark
Idle OS	Run guest OS with idle state (for both OS types)
CPU intensive task	For Linux Compiling XEN source code For Windows Installing Cygwin
Memory Intensive task	Playing video (for both OS types)
Network Intensive Task	Web server (for both OS types)

جدول2. معیارهای حجم کاری

1. حجم کاری در حالت بیکاری سیستم عامل: در یک سیستم عامل بیکار، صفحات حافظه خراب تولید شده، دارای کمترین زمان تولید در طول مهاجرت VM بدون اجرای هیچ گونه حجم کار سنگینی اند. تغییرات حافظه در صورت بیکار بودن سیستم عامل به منظور ارزیابی مدل مهاجرت زنده VM DSM استفاده می شود.

2. حجم کار فشردهCPU : کامپایل کردن کد منبع (VM لینوکس) و نصب و راه اندازی Cygwin (VM ویندوز) یک بار CPU سنگین بوده و در این موارد از حجم کار، زمان کامپایل در CPU بالاتر از دیگر وظایف پردازنده است.

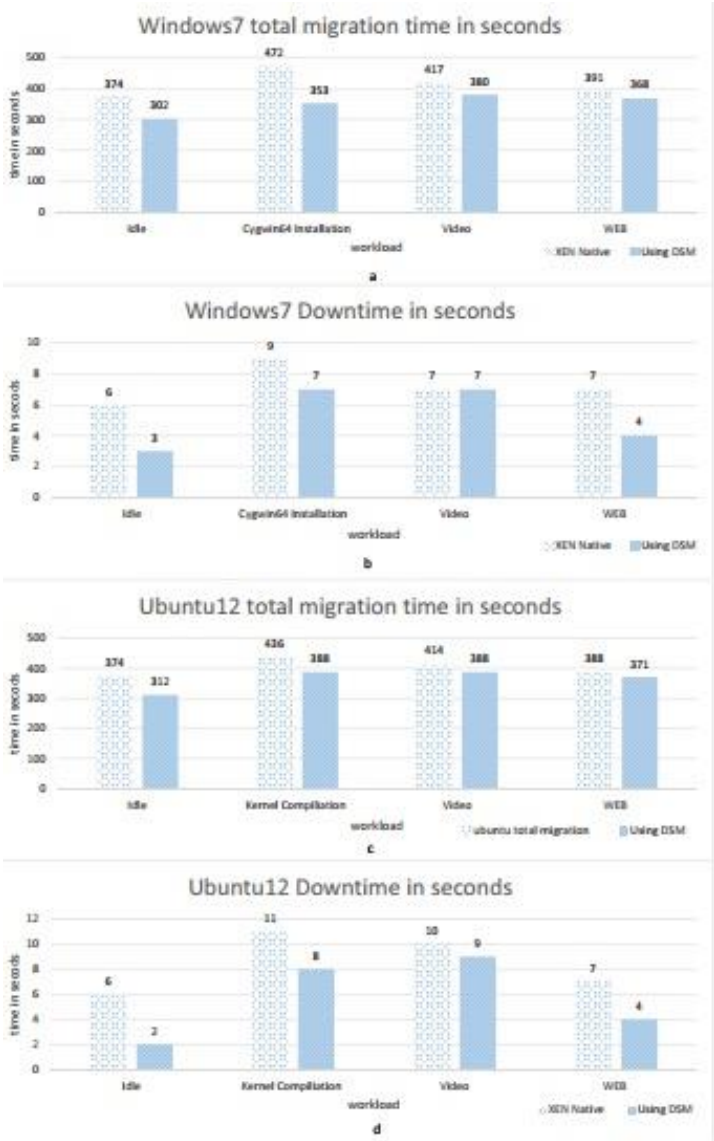
3. حجم کار فشرده حافظه: حجم کار فشرده حافظه با اجرای ویدئو در طول روند مهاجرت زنده برای هر دو نوع سیستم عامل، ویندوز و لینوکس ایجاد میشود، علاوه بر بار خواندن دیسک فشرده برای ویدئو از هارد دیسک.

4. حجم کار فشرده شبکه: برای وظیفه شبکه ، هر اتصال مشتری به وب سرور باید حفظ شود تا زمانی که مشتری به آن اتصال خاتمه دهد. داده های منتقل شده در وب عمدتا دارای یک حالت شبکه بزرگتر برای حفظ اتصال مشتریان به وب سرور هستند که باعث افزایش ردپای حافظه با ویژگی لوکالیتی بالاتر میشود.

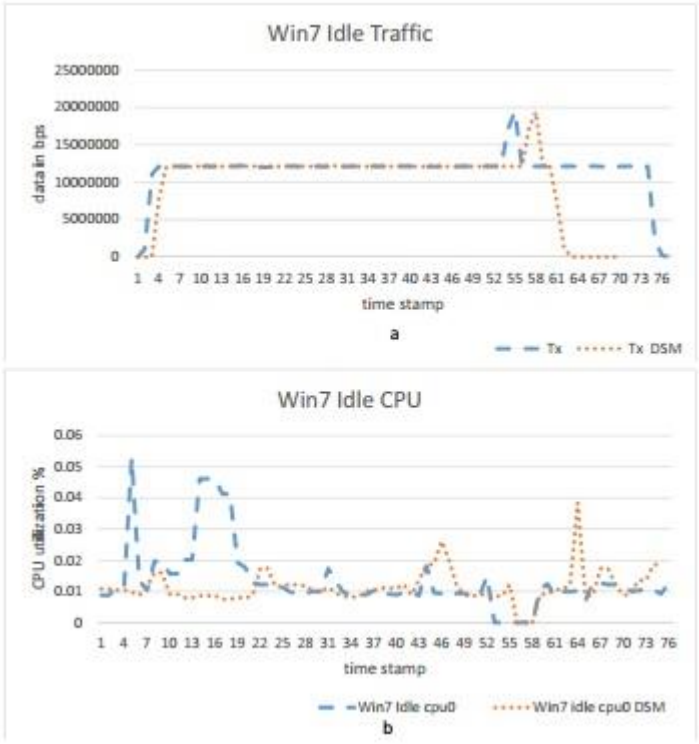
همانگونه که در شکل 4 الف نشان داده شده، مجموع زمان مهاجرت در تمام حجم کار کاهش یافته است. اما بهترین حالت آن با نصب Cygwin و حجم کار در حالت بیکار بوده که به ترتیب در حدود 25٪ و 20٪ افزایش است، در حالی که این افزایش در حجم کاری وب و ویدئو در حدود 6٪ و 8٪ بود. شکل 4 ج مجموع زمان مهاجرت را در VM لینوکس به تصویر می کشد. با DSM تمامی حجم کار کم بوده و بهترین نرخ در حالت بیکار بودن VM با 16.6٪ افزایش یافته است که مجموع زمان مهاجرت را کاهش داده است. از آنجا که ردپای حافظه لینوکس کمتر از ویندوز است، به روز رسانی صفحات خراب با لوکالیتی بالا در لینوکس کمتر از ویندوز می باشد. این باعث می شود تا DSM با ویندوز بهتر عمل کند.

در شکل 4 ب و 4 د، زمان خرابی بدست آمده برای ویندوز و لینوکس با DSM ، با یک کاهش عالی در تمام موارد حجم کار مواجه شد است. در حدود 42.8٪ کاهش زمان خرابی در حجم کاری وب برای هر دو سیستم عامل ویندوز و لینوکس بدست آمده است. در بازی های ویدئویی ویندوز هیچ افزایشی صورت نگرفته چرا که GPU همراه بار حافظه و چینش تصاویر ویدئویی کار میکند. اما در لینوکس به 10٪ افزایش میرسیم زیرا در لینوکس ویدئوها در یک راه هموار stream نمیشوند. DSM زمان خرابی را با کاهش تعداد صفحات معیوب پس از حالت تعلیق و ادامه کاهش میدهد. پردازنده های موردهدف می توانند تمام صفحات حافظه را به طور مستقیم مشاهده کنند.

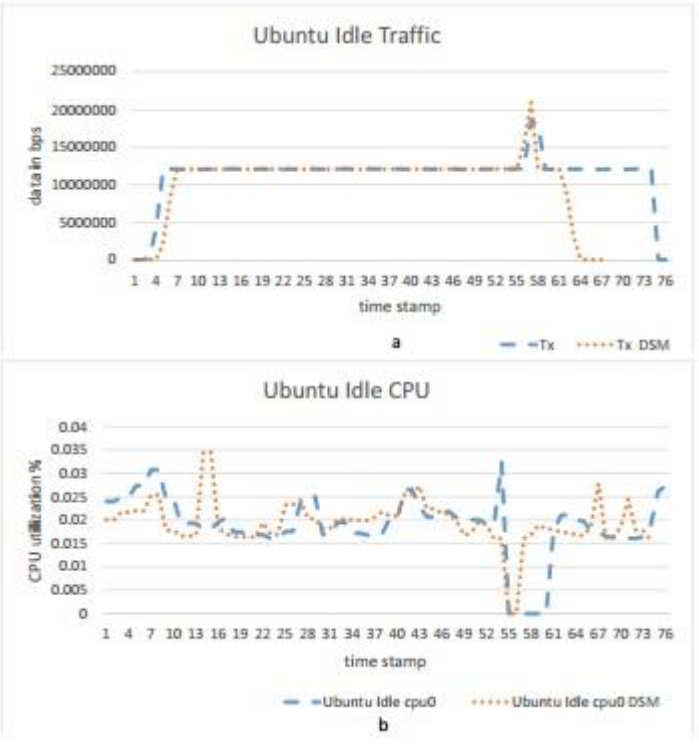
شکل های 5 تا 12 شبکه و عملکرد پردازنده VM را در طول مهاجرت زنده VM نشان می دهند. بازه زمانی به عنوان شکاف زمانی ثابت برای نشان دادن زمان شروع مهاجرت زنده VM ، زمان خرابی و زمان مهاجرت کل نرمالسازی شده است. به وضوح میتوان رابطه بین عملکرد پردازنده های VM و فعالیت ترافیکی را مشاهده کرد. در تمام موارد، ترافیک شبکه به یک پهنای باند ثابت Mbps12 برای محافظت از شبکه محدود شده و در طول دوره خرابی میزان ترافیک برای کاهش خرابی افزایش می یابد.



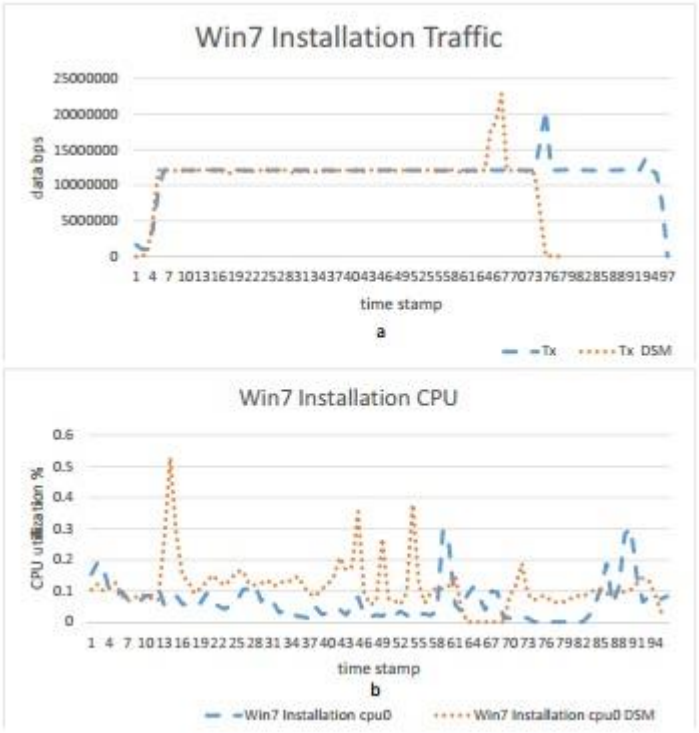
شکل 4. زمان خرابی و زمان مهاجرت کل



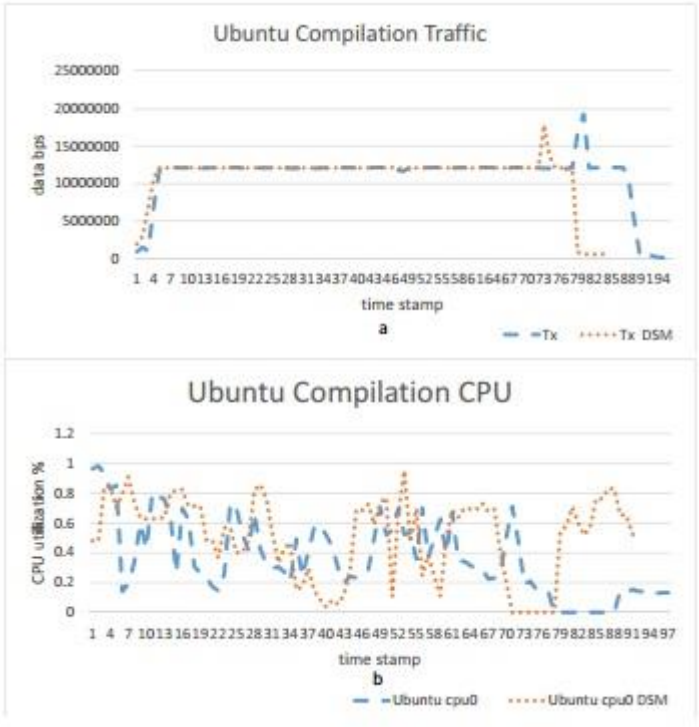
شکل 5. حجم کاری در حالت بیکاری ویندوز



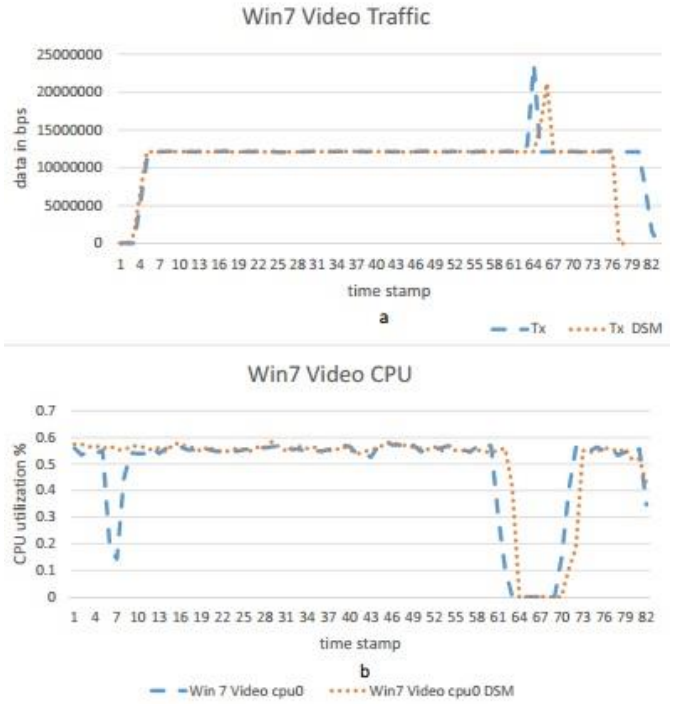
شکل 6. حجم کاری در حالت بیکاری لینوکس



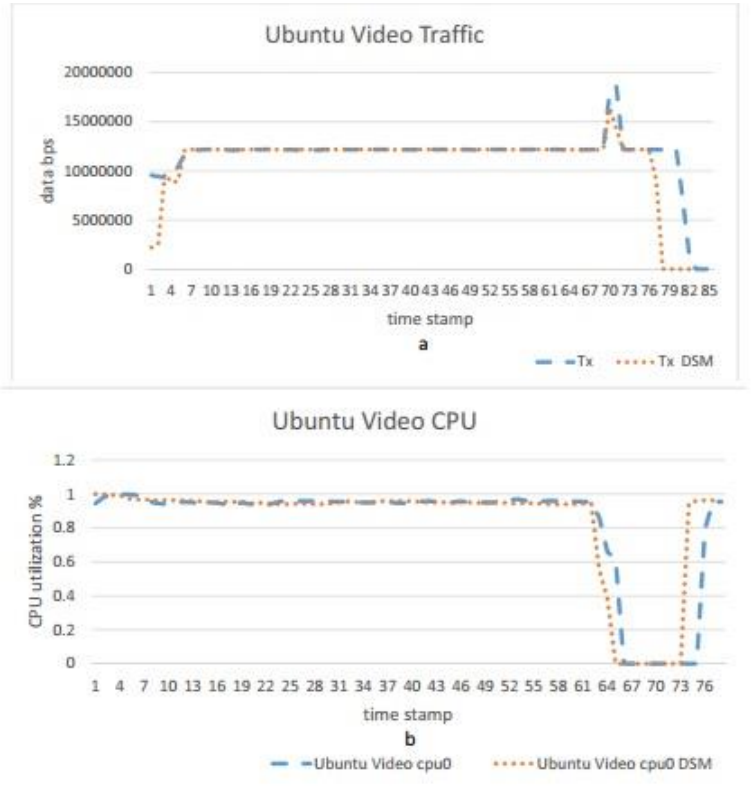
شکل 7. حجم کار فشرده پردازنده ویندوز



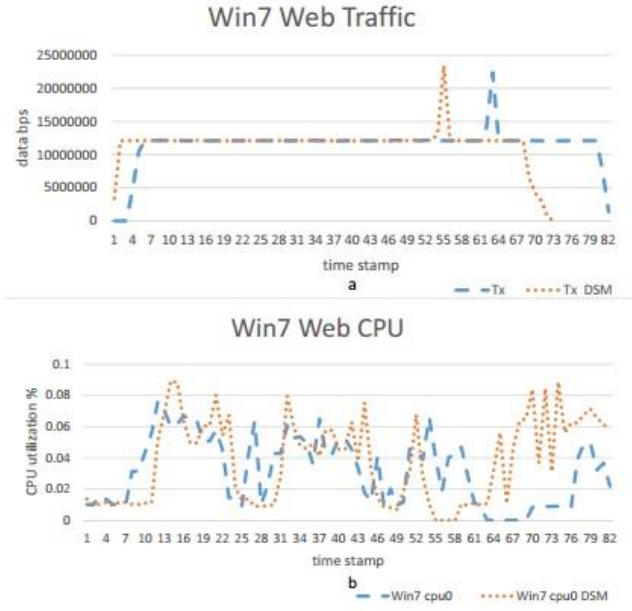
شکل 8. حجم کار فشرده پردازنده لینوکس



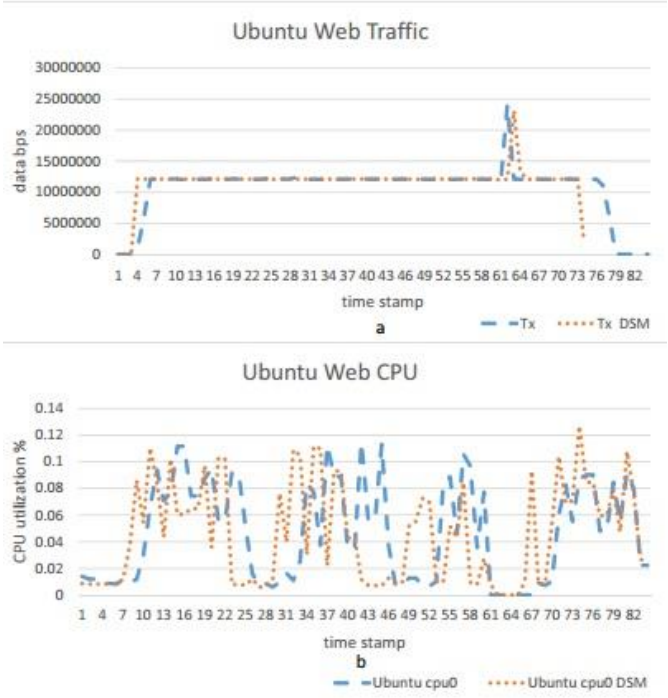
شکل 9. حجم کار فشرده ویندوز



شکل 10. حجم کار فشرده لینوکس



شکل 11. حجم کار فشرده شبکه ویندوز



شکل 12. حجم کار فشرده شبکه لینوکس

5. نتیجه گیری

روش نوینی برای اجرای مهاجرت زنده VM در مراکز داده ابری با استفاده از یک حافظه مشترک توزیع شده با خوشه محاسباتی با کارایی بالا ارائه شده است. به این ترتیب، حافظه هر گره محاسباتی به اشتراک گذاشته و در دسترس تمام گره های پردازنده با انتزاع بالا برای گره های حافظه محلی است. در مدل DSM از روش پیش نسخه مشابه ای استفاده شده است، به طوری که مرحله تکرار همان حالت پایدار حافظه را با انتقال حداقل صفحات خراب فراهم می کند، به دلیل ویژگی های محلیت DSM پیشنهادی، هیچ صفحه حافظه با تغییرات زیاد ارسال نمیشود. علاوه بر این، افزایش سرعت در ارسال صفحات حافظه و یا وضعیت پردازنده و شبکه با استفاده از موازی سازی خودکار نگاشت در دسترس پذیری حافظه بین دستگاه منبع و ماشین مقصد به دست آمده است. مدل ارائه شده با معماری مجازی با استفاده از ذخیره سازی مشترک ایجاد و یکپارچه شده است. کار پیشنهادی آینده ، ادغام خوشه DSM HPC با مهاجرت VM در یک واحد می باشد.

REFERENCES

[1] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, “Live migration of virtual machines,” Symposium on Networked Systems Design, pp. 273–286, 2005.

[2] J. Hai, D. Li, W. Song, S. Xuanhua, and P. Xiaodong, “Live virtual machine migration with adaptive, memory compression,” Conference on Cluster Computing and Workshops, pp. 1–10, 2009.

[3] X. Min, M. Vyacheslav, S. Jeffrey, V. Ganesh, W. Boris, and V. Inc, “Retrace: Collecting execution trace with virtual machine deterministic replay,” Workshop on Modeling, Benchmarking and Simulation, 2007.

[4] J. Nelson, B. Holt, B. Myers, P. Briggs, L. Ceze, S. Kahan, and M. Oskin, “Latency-tolerant software distributed shared memory,” USENIX Annual Technical Conference, pp. 291–305, 2015.