APJ 6(1):45-54, Spring 2025

Arman Process Journal (APJ)

Homepage: https://www.armanprocessjournal.ir

Resource Management Optimization Approach in Distributed Operating Systems

M. Mohammadi[1], R. Shiri[2*],

[1] Department of Computer Science, Islamic Azad University, Khodabandeh Branch, Zanjan, Iran
[2] Department of Computer Science, Islamic Azad University, Khodabandeh Branch, Zanjan, Iran

ABSTRACT

RESEARCH PAPER

Received: 7 December 2024
Accepted: 18 April 2025

KEYWORDS:
Distributed operating system,
Resource management,
Process management,
Scheduling,

[1] **Corresponding author:**
shiri.research @gmail.com.

Distributed operating systems are a very important part of the software design and implementation work for managing systems with a large number of processors. On the other hand, research in the areas of computer science research on distributed systems has increased significantly in the past few decades. One of the topics discussed in such operating systems is the discussion of resource management through process management. For process management in a distributed operating system, there are mechanisms that allow process transfer, download, remote debugging, state vectoring, and remote emulation of the operating system. Process management facilities can be realized by the distributed operating system kernel, and resource utilization improvement is possible through this. In this research, we intend to examine the evolution of distributed operating systems, how to manage processes, migration, clustering, how to transact messages, and other topics about distributed operating systems. We also consider an example of such operating systems called Amoeba, and we examine process management, inter-process communication, and its efficient mechanisms for improving resource management.

نشریه تخصصی آرمان پردازش، دوره ۶، شماره ۱، بهار ۱۴۰۴

فصلنامه تخصصی آرمان پردازش
(APJ)
Homepage: www.armanprocessjournal.ir

فصلنامه تخصصی فناوری اطلاعات و ارتباطات
شماره مجوز: ۸۷۰۹۰

رویکرد بهبود مدیریت منابع در سیستم عامل های توزیع شده

مهدی محمدی[۱]، رضا شیری[۲*]
گروه مهندسی کامپیوتر، دانشگاه آزاد واحد خدابنده، زنجان، ایران
گروه مهندسی کامپیوتر، دانشگاه آزاد واحد خدابنده، زنجان، ایران

چکیده

سیستم عامل های توزیع شده بخش بسیار مهمی از کار طراحی و اجرای نرم افزار برای مدیریت سیستم هایی حاوی تعداد زیادی پردازنده را تشکیل می دهند، از طرفی پژوهش ها در حوزه های تحقیقات علوم کامپیوتر پیرامون سیستم های توزیع شده در چند دهه گذشته به میزان قابل توجهی افزایش یافته است. یکی از موضوعاتی که در این گونه سیستم عامل ها مطرح می شود بحث مدیریت منابع از طریق مدیریت فرایند است که برای مدیریت فرایند در یک سیستم عامل توزیع شده مکانیزم هایی وجود دارد که امکان انتقال فرایند، دانلود، اشکال زدایی از راه دور، حالت برداری و شبیه سازی راه دور سیستم عامل را ایجاد می کند. امکانات مدیریت فرایند توسط هسته سیستم عامل توزیع شده قابل تحقق است و بهبود کاربرد منابع از طریق این مهم ممکن می گردد. در این پژوهش قصد داریم بر سیر تکامل سیستم عامل های توزیع شده، نحوه مدیریت فرایند، مهاجرت، خوشه بندی، نحوه تراکنش پیام و سایر مباحث پیرامون مدیریت بهینه منابع در سیستم عامل توزیع شده، همچنین یک نمونه از این گونه سیستم عامل ها به نام آموبا (آمیب) را مد نظر قرار داده و مدیریت فرایند، ارتباطات درون فرایندی و مکانیزم های کارآمد بودن آن را در خصوص ایجاد دیدگاه بهبود مدیریت منابع مورد بررسی قرار می‌دهیم.

واژگان کلیدی:
سیستم عامل توزیعی،
مدیریت منابع ،
مدیریت فرآیند،
زمانبندی،

مقدمه

روند تکامل و پیشرفت کامپیوترها در چند دهه اخیر به گونه ای بوده است که ماشین های بزرگ و یکپارچه که اجازه دسترسی کامل آن تنها به یک کاربر داده می شد جای خود را به ماشین هایی داده است که به کاربران متعددی اجازه دسترسی هم زمان را می دهند. با مرور زمان و پیشرفت معماری کامپیوترها، ماشین ها کوچکتر شدند و مجددا به کامپیوترهای تک کاربره تبدیل گردیده و کامپیوتر شخصی یا (PC) نامیده شدند و نهایتا تا به امروز با متصل نمودن این کامپیوترهای شخصی به یک سیستم مرکزی به منظور استفاده از دیسک ها، پرینتر ها، پردازنده ها و رایانش توزیع شده به وجود آمده است. از اواخر دهه ۱۹۷۰ میلادی تحقیقات پیرامون رایانش توزیع شده، علی الخصوص سیستم عامل های توزیع شده میزان قابل توجهی از پژوهش ها در سطوح عالی را به همراه داشته و در حال حرکت سریع به سمت حوزه های تحقیقات علوم کامپیوتر در محیط دانشگاه می باشد. در دهه های پیش رو انتظار می رود هزینه تراشه ها CPU به کاهش خود ادامه داده و نهایتا منجر به سیستم هایی شود که حاوی تعداد زیادی از پردازنده ها می باشند و طراحی و اجرای نرم افزار برای مدیریت فرایندها و استفاده از تمام نیروی رایانش آن ها بخشی بسیار مهم محسوب می گردد که این امکانات جهت مدیریت فرایند توسط هسته سیستم عامل توزیع شده قابل تحقق است و آن ها را می توان توسط سرویس های فضای کاربری استدلال نمود: سرویس اشکال زداء سرویس توازن بار، سرویس نمونه سازی یونیکس، سرویس بازرسی و ... [۱]. سیستم عامل های توزیع شده مختلفی را می توان نام برد مثلا K۲ یک نمونه از این گونه سیستم عامل های توزیع شده ی یکپارچه است که معماری آن به گونه ای است که مشکلات مدیریت منابع در شبکه های ناهمگن را مرتفع کرده و کاربرد توزیع شده دارد. اما یکی از مطرح ترین سیستم عامل های توزیع شده سیستم عامل آموبا یا آمیب است. این سیستم عامل با عملکردی بالا طراحی گردیده جهت تعامل بین کلاینت و سرور با استفاده از مدل RPC به معنای از مدل RPC رویه فراخوانی از راه دور. در سال ۱۹۹۰ در دانشگاه Vrije آمستردام گروهی تحت هدایت پروفسور تنن باوم گرد هم آمدند و تاکنون در مورد سیستم عامل های توزیع شده تحقیق می کنند. نتیجه این تحقیقات سیستم عامل توزیع شده آموبا است که برای محیط هایی با تعداد کامپیوتر زیاد طراحی شده است البته این سیستم عامل برای دانشگاه ها و موسسات تحقیقاتی رایگان است. آموبا یک سیستم عامل چند منظوره توزیع شده است. این سیستم عامل می تواند از چندین ماشین بهره بگیرد و با آن ها طوری رفتار کند که انگار یک سیستم واحد و یکپارچه است. به طور کلی، کاربران از تعداد محل و پردازنده هایی که برای اجرای دستورات وی به کار گرفته می شود، مطلع نیستند. همچنین اطلاعات محل و سرورهای فایلی که فایل های آن ها در آن ذخیره می شود نیز نامشخص است. از دیدگاه کاربر، آموبا درست همانند یک سیستم عامل سنتی اشتراک زمانی است. آموبا یک پروژه ناتمام است و هنوز در مرحله تکمیل به سر می برد. این سیستم عامل به عنوان بستری برای تحقیق و توسعه کد در سیستم های موازی و زبان ها و پروتکل های مرتبط با آن به کار می آید [۲]. همچنین این سیستم می تواند یونیکس را شبیه سازی کند و ظاهری درست همانند یونیکس دارد اما به عنوان جایگزینی برای یونیکس نیست و تنها عملکردی مشابه دارد. این سیستم عامل برای دانشجویان و محققانی که می خواهند کد منبع سیستم عامل را مشاهده و طریقه عملکرد آن را از نزدیک ببینند، بسیار مناسب است. آموبا همچنین برای برنامه نویسی به روش توزیع شده (چند کاربر به طور مجزا روی چند پروژه مختلف کار کنند) و سیستم های موازی (یک کاربر از ۵۰ پردازنده استفاده کند تا جیشطرن را به طور موازی بازی کند) استفاده می شود. البته ایجاد نرم افزارها برای این سیستم عامل ساده است، همانند دستور make در یونیکس، یک دستور به نام amake وجود دارد [۳].

مدیریت منابع در سیستم عامل های توزیعی یعنی مجموعه ای از کامپیوترهای محاسبه گر و خود مختار که هر کدام می توانند چندین کاربر داشته باشند و برای کاربران به نظر می رسد که این مجموعه کامپیوترها واحد هستند و ، این سیستم ه مگرا است، برخی منابع در مورد توزیع شدگی سیستم عامل

توزیع شده بیان می کنند که یک سیستم عامل توزیع شده مجموعه ای است از کامپیوترهای مستقل که کاربران آن را به صورت یک سیستم واحد می بینند. به عنوان نمونه یک معماری توزیع شده برای سیستم عامل های چند هسته ای پیشنهاد شده که مزیت اصلی آن در دسترس بودن منابع می باشد که در پروژه های صنعتی و تحقیقات و در سطح شرکت ها برای صدور گواهینامه ایمنی برنامه های کاربردی استفاده می شود [۴]. آموبا سیستمی توزیع شده است و در آن چندین ماشی ن که به یکدیگر متصل هستند را کنترل و مدیریت می کند. نیازی نیست که این ماشین ها همه از یک نوع باشند. این ماشین ها می توانند در یک شبکه LAN به یکدیگر متصل شوند. آموبا از شبکه پروتکل قدرتمند FLIP استفاده می کند. اگر یک ماشین آموبا بیشتر از یک رابط شبکه داشته باشد، به طور خودکار به عنوان مسیریاب بین چند شبکه به کار گرفته خواهد شد شبکه و های LAN مختلف را به یکدیگر متصل خواهد گردد.

استفاده از یک سیستم عامل اجازه می دهد تا زمان اجرا با مدیریت فرایندهای کاربر، پردازش ها مدیریت شود که این کار می تواند استفاده از سیستم را کارآمدت ر کند، در طراحی های اولیه مدیریت فرایند در سیستم عامل های توزیع شده مکانیزم ، هایی لازم است که بتوانند کارایی طرح های اولیه مدیریت فرایند در سیستم عامل های همه منظوره موجود را افزایش دهند .قابلیت افزوده شده باید با ویژگی های انواع سیستم های توزیع شده خود را سازگار کنند: ایستگاه های کاری شخصی، ماشین های سرور مشترک و سیستم های میزبان متصل توسط یک شبکه محلی سریع. در حالت عادی ایستگاه های کاری توسط یک فرد مورد استفاده قرار می گیرند اما اگر هیچکس از آن ها استفاده نکند تبدیل می شوند به منبع محاسباتی برای سایر کاربران و ایستگاه ها. ماشین های با سرور مشترک هم یک سیستم فایل توزیع شده اند و دارای قابلیت هایی مثل اینترنت، دسترسی به پرینتر، درایوهای نواری و ... هستند. منظور از سیستم های میهمان نیز سیستم عامل های سنتی است که به کمک اشتراک گذاری نرم افزار به سیستم توزیع شده متصل می شوند.

مدیریت فرایند در یک سیستم توزیع شده

اگر بخواهیم یک سیستم توزیع شده ی همه منظوره با زبان های مختلف، فایل سیستم های مختلف و نرم افزار ها و سخت افزارهای مختلف ایجاد کنیم محیط ما یک محیط برنامه ریزی ناهمگن است و روند طراحی باید به گونه ای باشد که امکان اجرای نرم افزار های موجود را فراهم کند. در این محیط برنامه های کاربران مختلف به طور مداوم یک پردازنده فیزیکی را به اشتراک می گذارند، بنابراین آن ها باید در فضاهای آدرس جداگانه اجرا شوند. از آنجا که تمام ماشین ها سیستم فایلشان محلی نیست طبیعتا برنامه ها باید از روی شبکه موجود دانلود شوند و این کار چند ثانیه طول می کشد. مکانیزم ارتباطی برنامه های کاربردی توزیع شده، تراکنش پیام است، یک جفت پیام: یک درخواست پیام از یک فرایند سرویس گیرنده به یک سرور است که به دنبال آن یک پیام پاسخ از سرور به سرویس گیرنده فرستاده می شود. در راس آن، فراخوان روال راه دور وجود دارد. هنگامی که با دقت طراحی و اجرا شود، تراکنش پیام از یکی از کارآمدترین پروتکل های ارتباطی را برای شبکه های محلی، هم از نظر تاخیر و هم از نظ ر توان عملیاتی، تشکیل می دهد [۵].

هنگامی که یک فرایند سرویس گیرنده درخواستی ارسال می کند، تا زمان رسیدن پاسخ مسدود می کند؛ هنگامی که سرور درخواست طلب کند، تا زمان دریافت یک درخواست، مسدود می کند. استفاده از تراکنش های پیام، پیامدهای متعدد ی برای طراحی محیط برنامه نویسی دارد. ابتدا، فرایندها هر تراکنش پیام را مسدود می کنند. بنابراین، جا به جایی سوییچ ها رخ می دهد: یکی زمانی که فرایند، فرایند دیگر را مسدود می کند و دیگری بعد از اینکه فرایند دوباره برای اجرای اصلی غیرمسدود می شود. اگر تراکنش های پیام بسیار سریع انجام شوند، سوییچینگ فرایند نیز بهتر است که سریع باشد . دوم، تراکنش های پیام موازی انجام نمی شود: هنگامی که سرویس گیرنده اجرا می شود، سرویس دهنده منتظر

درخواست می ماند و هنگامی که سرویس دهنده اجرا می شود، سرویس گیرنده منتظر پاسخ می شود . تنها یک فرایند در آن واحد اجرا می شود، البته روی ماشین های مختلف. از یک روش می توان برای اجرای تراکنش های غیرمسدود استفاده نمود: سوییچ فرایند نیازی به رقابت با تراکنش های پیام در سرعت ندارد و عمل موازی سازی را می توان با ارسال درخواست به همه سرویس دهنده ها به طور هم زمان انجام داد. با این حال، در این روش با یک مجموعه مشکلات جدید مواجه می شویم. در بسیاری از سیستم های توزیع شده برای تسهیل نوشتن نرم افزاری که از سیستم موازی استفاده می کند، از فرایندهای سبک وزن و مسدود سازی تراکنش های پیام استفاده می شود. برای مهاجرت فرایندها در سیستم های توزیع شده، مکانیزم هایی استفاده نماید. با توجه پیاده سازی یا پیشنهاد شده است اما تا کنون الگوریتمی پیشنهاد نشده که از مهاجرت برای توازن بار به زمان مورد نیاز برای مهاجرت یک فرایند گسترده (به ترتیب ده ثانیه ،) مهاجرت برای توازن بار به نظر چندان سودمند نمی آید [۶].

سیستم عامل توزیع شده آموبا

آموبا یک سیستم عامل توزیع شده مبتنی بر الگوی ارتباط فرایندهای سرویس گیرنده با سرویس ها از طریق تراکنش های پیام است. این سیستم عامل از قابلیت هایی برای دسترسی به سرویس ها و اشیایی که این سرویس ها اجرا می کنند، استفاده می .کند یکی از این قابلیت ها مرجع ۲۵۶ بیتی برای یک شئ است که ۶۴ بیت اول پورت است و اشاره به مدیریت سرویس توسط شی دارد؛ ۶۵ بیت بعدی برای استفاده سیستم به عنوان مکان در دسترس است؛ ۱۲۸ بیت باقیمانده برای شناسایی شئ به سرویس تخصیص می یابد. یکی از قابلیت هایی که به این شیوه تولید شده است و شامل بین های کافی است، این احتمال وجود دارد که یک کاربر غیرمجاز حدس بزند که می توان از یکی از قابلیت های اهداف چشم پوشی کرد. از این قابلیت ها برای محافظت استفاده می شود و همچنین به عنوان مکانیزم اولیه برای رسیدگی به د رخواست ها تا عملیات و اهداف انجام شوند .هنگامی که یک سرویس گیرنده درخواستی را می فرستد، سیستم از پورت برای تعیین نوع سرویسی که باید درخواست را اداره نماید، استفاده می کند. یک سرویس دهنده نیز از طریق عملیات موقعیت یابی پیدا می شود، یعنی از طریق انتشار بست ه های (شما کجا هستید .) سرویس دهنده از بخش خصوصی این قابلیت برای شناسایی شئ استفاده می کند. بعد از رسیدگی به یک درخواست، سرور پاسخ می دهد [۷]. بیشتر سرویس ها در فضای کاربری اجرا می شوند. هسته آمیب تنها یک سرویس حداقل ایجاد می کند: امکان تراکنش پیام، مدیریت فرایند و مسیر دسترسی به تجهیزات جانبی. برای مثال، سرویس فایل یک سرویس فضای کاربری بدون امتیازات ویژه است به جز دانش قابلیت های لازم برای گرفتن دیسک هایی که در آن ها فایل ها ذخیره شده اند. تراکنش های پیام مسدود شده اند و سیستم بافری ایجاد نمی کند. هنگامی که ورسر فرمان getrequest را فراخوانی می کند (پورت، قابلیت، بافر درخواست (،) پورت سرور سیستم را شناسایی می کند ،) سرور مسدوی می شود تا اینکه یک درخواست از راه برسد. سرور با فرمان putreply یک پاسخ باز می فرستد که مسدود نشود. هنگامی که سرویس گیرنده فرمان trans را فراخوانی نماید، تا زمانی که پاسخ از سمت سرور برسد، مسدود می شود [۸].

سرور کرنل (هسته)

هسته یا کرنل آموبا قطعات حافظه و ارتباطات درون فرایندی را مدیریت می کند و از فرایندهایی که شامل چند نخ هستند پشتیبانی می کند. این هسته برای تحقق انتزاع فرایند در آمیب با سه شئ اصلی سروکار دارد .یک خوشه یک فضای آدرس دهی مجازی است که شامل تعدادی قطعه و تعدادی موضوع کنترل به نام وظایف است. علت مشترک بودن فضای آدرس میان وظایف، بهره وری است: وظایف می توانند اطلاعات را در یک حافظه مشترک، بین یکدیگر به صورت کارآمدتری مبادله نمایند و از آنجایی که وظایف زمینه کمی د ارند، سوییچ کردن وظیفه می تواند خیلی سریع اتفاق افتد. از مفهوم اتصالات بین وظایف در سیستم های توزیع شده

مدرن زیادی استفاده شده است که در بین این سیستم عامل ها قابل توجه هستند [۹].

قطعه ها

یک قطعه یک بخشی خطی از حافظه است. قطعه یک شئ است که توسط سرویس کرنل مدیریت می شود. قطعات با نگاشت اشیاء در فضای آدرس یک خوشه با استفاده از فراخوان سیستم seg_map ایجاد می شوند. این امکان وجود ندارد که یک قطعه موجود را نگاشت کرد؛ بنابراین، قطعات را نمی توان میان شهخو ها به اشتراک گذاشت. فراخوان برای مدیریت قطعه در شکل نشان داده شده است.

System Calls
sid = seg_map(segment, address, lengh, how)
seg_unmap(sid)
seg_grow(sid, newlength)
seg_info()
Transactions
Seg_Create(kernelcap, incap, outcap)
Seg_Read(capability, offset, buffer, count)
Seg_Write(capability, offset, buffer, count)
Seg_Length(capability)
Seg_Delete(capability)

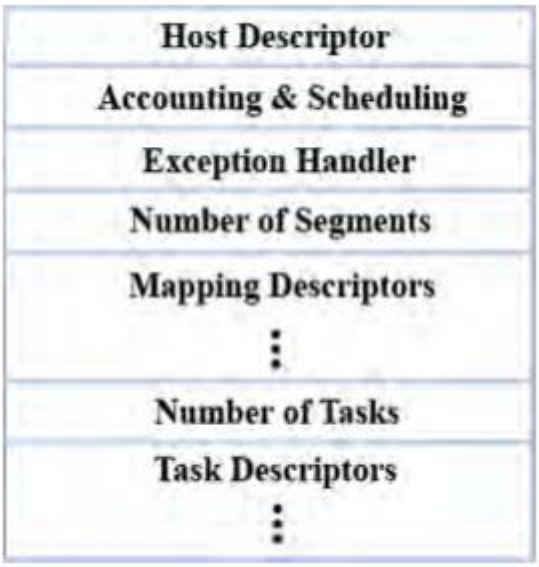
شکل۱. تراکنش ها و فراخوان های سیستم مدیریت قطعه

تفاوت اساسی میان تراکنش ها و فراخوان های سیستم، این است که فراخوان های سیستم توسط هسته محلی اداره می شود و بنابراین، هسته محلی تنها برای کنترل اشیاء محلی قابل استفاده است. با این حال، تراکنش ها برای سرویسی رسیدگی می شوند که ممکن است محلی یا راه دور باشند، تراکنش Read_Seg بنابراین برای خواندن محتویات قطعات راه دور استفاده می شوند. با این حال، تراکنش ها باید به سرویسی آدرس دهی شوند که ممکن است محلی یا راه دور باشد، بنابراین تراکنش Read_Segرا می توان برای خواندن محتویات قطعات راه دور به کار برد. فراخوانی map_segیک قطعه جدید ایجاد می کند که آن را به محتویات قطعه مقداردهی می نماید که با segment نشان داده می شود و این خود یک توانایی محسو ب می شود . howگزینه های نگاشت مثل نگاشت فقط خواندنی را مشخص می کند که تحت تاثیر فراخوان grow_Seg و ... قرار دارد. این فراخوان یک مقدار صحیح کوچک به نام شناسه قطعه را باز می گرداند که قطعه نگاشت شده را نشان می دهد. این عدد صحیح در فراخوان های unmap_seg و grow_segمورد استفاده قرار می گیرد. هنگامی که یک قطعه از نگاشت خارج می شود، از فضای آدرس یک خوشه خارج می گردد اما به صورت یک شئ به حیات خود در حافظه ادامه می دهد unmap_Seg .قالبیت این شئ حافظه را برای دستکاری بیشتری با تراکنش ها جهت مدیریت قطعه باز می گرداند info_Seg .اطلاعات مربوط به فراخوانی نگاشت فعلی حافظه خوشه را باز می گرداند. تراکنش های مدیریت قطعه برای خودشان کار می کنند.

خوشه ها

سیستم عامل های توزیع شده چند ویژگی اصلی دارند ازجمله: تفکیک کاربردی بر اساس قابلیت ها و خدمات و توانایی ها و هدف و هویتشان ، توزیع ذاتی مثل اطلاعات متفاوت توسط افراد مختلف که ایجاد و حفظ شده است ، قابلیت اعتماد یعنی بک آپ دراز مدت در محل های مختلف دارند ، مقیاس پذیری به معنای قابلیت سیستم برای افزایش منابع و اقتصادی بودن یعنی به اشتراک گذاری منابع با هویتهای بسیار برای کمک به کاهش هزینه مالکیت . هویت های متعدد در یک سیستم توزیع شده می توانند به صورت هم زمان و احتمالا به صورت خودگردان (مستقل) فعالیت کنند. وظایف به طور مستقل انجام شده اند و می باشند شکست و ها مستقل اقدامات توسط تبادل پیغام ها هماهنگ شده اند. همچنین، هویت ها هتروژن (ناهمگن) هستند. به طور کلی ، هیچ فرایند یا هویت مجزا وجود ندارد که دارای دانش کل وضعیت سیستم باشد [۱۰].

همچنین سیستم های توزیع شده اهداف مشخصی را دنبال می کنند از جمله : هتروژنیتی یا همگنی یعنی وجود تعامل علیرغم تفاوت های سخت افزاری ، برنامه نویسی ، فرمت ها و ...، شفافیت یعنی مخفی ماند ن پیچیدگی ها از دید کاربر، مدیریت شکست یعنی توانایی در برابر اختلالات، مقیاس پذیری یعنی قابلیت افزایش منابع، هم زمانی یعنی دسترسی مشترک به منابع، قابلیت توسعه نقل، مکان و تعادل بار، وجود امنیت در کل سیستم و مسائل دیگری که موسسات تحقیقاتی و شرکت ها مطرح نموده اند به همین دلیل با توجه به اینکه تولید برنامه های کاربردی برای سیستم ها از جنبه های مختلفی مانند مدیریت فرایند، ارتباطات توزیع شده در سیستم و قدرت تحمل خطا بسیار پیچیده است، می توان گفت تولید برنامه های کاربردی برای این سیستم عامل ها نیز کار دشواری است.



شکل۲. توصیف کننده خوشه

توصیف کننده میزبان، نوع پردازنده ای خوشه در آن اجرا می شود را توصیف می نماید. ورودی های آن دارای یک نوع و مقدار است. برای مثال، در نوع "set instruction" ، "مقادیری مثل VAX ، M68000یا NS32000 فرض می شوند و مجموعه دستورالعمل هایی توصیف می شوند که گر ه خوشه به آن تعلق دارد. از نوع گزینه های مستقل از مجموعه دستورالعمل برای نشان دادن اینکه خوشه نیاز به گزینه های مجموعه دستورالعمل ها دارد یا خیر استفاده می شود، مثل نقطه اعشاری یا مجموعه دستورالعمل های پیشرفته. نوع اندازه حافظه دارای مقداری است که نشان دهنده حداکثر اندازه ای است که فضای آدرس خوشه برای رشد ممکن است به آن نیاز داشته باشد. انواع احتمالی دیگری نیز وجود دارند؛ انواع جدید را به سادگی می توان افزود. سرویس کرنل انواع مفید را تشخیص می دهد و از مقادیر آن ها برای تعیین اینکه آیا می تواند خوشه را اداره کند یا نه استفاده می نماید. انواع دیگر را می توان توسط سرویس های کاربر مورد استفاده قرار داد که توصیف کننده های خوشه را دستکاری می کنند (مثل، سرویس نمونه سازی) زمینه اداره استثناء باعث می شود که پورت سرویس در هنگام وجود استثناء ها آن ها را اداره کند. سپس به دنبال mapping descriptors، برای هر قطعه فضای آدرس خوشه وجود دارد. این هسته با انجام فراخوان های map_seg که توسط هر یک از توصیف کننده های نگاشت مشخص شده است، فضای آدرس مجازی خوشه را ایجاد می کند . درنهایت، فهرستی از توصیف کننده های وظیفه وجود د ارد، برای هر وظیفه در خوشه، وضعیت هر وظیفه در خوشه را نشان می دهد. این یکی از مزایای داشتن تراکنش ها به عنوان تنها راه برای برقراری ارتباط با خارج از خوشه است: کرنل آمیب حالت کمی برای وظایف حفظ می کند. حالت یک وظیفه شامل مواردی مثل قابل اجرا بودن یا مسدود شدن در یک نشانه یا شرایط متغیر، مقدار شمارنده برنامه، اشاره گر پشته، حرف وضعیت پردازنده و ثبات های دیگر است و اگر یک تراکنش در حال انجام باشد، شامل وضعیت آن نیز می شود. در هر لحظه، یک وظیفه می تواند تنها شامل دو تراکنش باشد: در حال انجام یک تراکنش با یک سرور باشد و در عین حال، به درخواست سرویس گیرنده نیز بپردازد [۱۱].

بیشتر فرایندها به عنوان رویدادی از اجرای یک برنامه روی یک کامپیوتر توسط یک مفسر فرمان ایجاد و مدیریت خواهند شد اما فرایندهای دیگر ممکن است موارد جدیدی ایجاد

کنند. تنها به قابلیتی نیاز است که امکان برقراری ارتباط با کرنل آموبا را بدهد. بیشتر کاربران برای اجرای کرنل آموبا روی ایستگاه کاری خود به قابلیت ایجاد خوشه دسترسی خواهند داشت؛ به همین دلیل کاربران می توانند فرایندهای جدیدی روی ایستگاه کاری مخصوص به خود ایجاد کنند. توانایی های مربوط به ایجاد فرایندها روی پردازنده مخزن، معمولا توسط یک سرویس "مخزن پردازنده" نگه داشته می شود که به عنوان یک عامل برای اجرای برنامه ها از جانب پردازش های کاربر عمل می کند. توازن بار را می توان توسط سرویس مخزن پردازنده انجام داد یعنی در زمانی که پردازنده های مخزن به طور صحیح تخصیص داده شده باشند [۱۲].

مهاجرت

اگرچه خوشه ها بعد از راه اندازی، به ندرت به یک میزبان جدید حرکت می کنند، مهاجرت یک مفهوم مرکزی در مکانی زم های مدیریت فرایند آمیب محسوب می شود. دلیل این است که بارگذاری خوشه های جدید در حافظه، گرفتن رونوشت های هسته ای، ایجاد نقاط بازرسی و انجام اشکال زدایی راه دور، همگی شبیه مهاجرت یک خوشه هستند. در واقع، اگر بتوانیم یک خوشه را از یک ماشین به ماشین دیگر مهاجرت دهیم، دانلود، حالت برداری، اشکال زدایی و ... می توانند ساده باشند. متعادل سازی باز توسط جابه جایی خوشه یکی از حوزه هایی است که به خوبی درک نشده و نمی دانیم که آیا واقعا با نوع فعلی ایستگاه های کاری شبکه و ها میسر واقع می شود یا خیر. برای مثال، مهاجرت یک خوشه ۵ مگابایتی حداقل ۷ ثانیه زمان نیاز دارد، چون برای کپی محتویات حافظه در یک اترنت ۱۰ مگابیتی به این زمان نیاز داریم؛ برنامه های ۵ مگابایتی اصلا غیرعادی نیستند، به ویژه که برای مهاجرت انتخاب شده باشند: خوشه هایی با عمر طولانی طول بزرگی دارند. بنابراین، مهاجرت نسبتا گران است و سود عملیات مهاجرت نیز باید زیاد باشد، با وجود این، مهاجرت می تواند مفید باشد. هنگامی که مالک یک ایستگاه کاری در شب خارج می شود، ایستگاه کاری می تواند خود را به یک پردازنده مخزن تبدیل و برای بقیه سیستم یک سرویس اجراکننده پردازش محسوب شود. هنگامی که مالک ، صبح باز می گردد و دوباره وارد می شود، خوشه های مهمان که در حال اجرا هستند می توانند به ایستگاه دیگری منتقل شوند . هسته ها مکانیزم مهاجرت خوشه را اجرا می کنند. آن ها یک خط مشی را پیاده سازی نمی کنند؛ تصمیم گیری برای انتقال یک خوشه و جایی که آن خوشه باید منتقل شود، در سطوح بالاتر سرویس انجام می شود [۱۳].

فرایندی که انتقال را دستور می دهد به سرور فرایند معروف است. هنگامی که یک خوشه از یک ماشین به ماشین دیگر می رود، هسته ماشین قدیمی باعث می شود که محتویات حافظه و توصیف کننده خوشه برای هسته ماشی ن جدید در دسترس قرار گیرد. هسته موجود در ماشین جدید خوشه را در حافظه بارگذاری می نماید و آن را راه اندازی می ما. کند به این دو هسته، میزبان قدیمی و میزبان جدید می گوییم، فرایند می تواند همه چیز را برای کنترل سیگنال های خوشه به صورت اشکال زدایی خوشه راه اندازی نماید . ابتدا، سرور فرایند یک سیگنال به خوشه ارسال می کند که باعث می شود میزبان قدیمی در مسیرهای خود منجمد شود و یک توصیف گر خوشه به سرور فرایند ارسال می کند (سرور فرایند برای این خوشه مثل آشکارساز خطا رفتار می کند). سپس، سرور فرایند توصیف گر خوشه را در یک درخواست RunCluster به میزبان جدید می فرستد . هنگامی که میزبان جدید درخواست RunCluster را دریافت می کند، قطعات لازم را تولید می کند و با ارسال درخواست های SegRead به میزبان قدیمی آن ها را مقداردهی می کند و آن ها را در فضای آدرس خوشه جدید نگاشت می کند. با سرویس گیرنده (میزبان جدید) و سرویس دهنده (میزبان قدیمی) می توان به طور مستقیم حافظه نگاشت شده را ارسال و دریافت نمود؛ بنابراین محتویات حافظه یک خوشه باید با سرعت بالای نصف یک مگابایت در ثانیه، روی اترنت کپی شود. هنگامی که تمام محتویات قطعه کپی شد، میزبان جدید خوشه را شروع و به سرور فرایند پاسخی شامل قابلیت خوشه جدید را ارسال می کند. بعد از آن، سرور فرایند با یک

درخواست DeleteCluster به میزبان قدیمی، خوشه قدیمی را حذف می کند . در حالی که عمل مهاجرت انجام می شود، خوشه در میزبان قدیمی در حالت منجمد قرار دارد . بنابراین، هسته به تمام پیام ها برای خوشه منجمد با یک پیام " دوباره تلاش کنید، این خوشه منجمد است" پاسخ می بعد. دهد از حذف خوشه، آن پیام ها دوباره در برخی نقاط ظاهر می شوند و هسته نیز با این پیام پاسخ می دهد: " این پورت برای این آدرس ناشناخته است". سپس، فرستنده عملیات مکان یابی انجام می دهد تا محل جدید خوشه را پیدا کند و ارتباط دوباره برقرار خواهد شد [۱۴].

نتیجه گیری

در این پژوهش با بررسی روند تکامل سیستم عامل های توزیع شده و چگونگی مدیریت بهینه منابع و فرایند در این سیستم عامل های توزیعی و همچنین یک نمونه سیستم عامل توزیع شده به نام آموبا یا آمیب مد اهداف، معماری و مدل های مختلف آن، مدیریت فرایند و ارتباطات درون فرایندی در آن را بررسی نموده و به این نتیجه رسیدیم که میتوان مکانیزمی ایجاد نمود تا بهبود مدیریت منابع یه منظور تحقق دانلود ، مهاجرت ، کنترل استثناها ، حالت برداری ، شبیه سازی و اشکال زدایی کارآمد باشد . طی ۲۰ سال گذشته گرایش به سمت شبکه های پر سرعت تر، سیستم های توزیع شده و معماری کامپیوتر های چند پردازنده ای، نشان می دهد که در آینده استفاده از سیستم های توزیع شده و نیاز به سیستم عامل های توزیع شده افزایش یافته و این سیستم عامل ها به شکل چشمگیری درکاربردهای مختلف مورد استفاده قرار بگیرند.

مراجع

[1] Pérez, H., Gutiérrez, J.J., Peiró, S., & Crespo, A. (2017, January). Distributed architecture for developing mixed-criticality systems in multi-core platforms. Journal of Systems and Software, 123, 145-159.

[2] Hollis, S.J., Ma, E., & Marculescu, R. (2016, October). NOS: A nano-sized distributed operating system for many-core embedded systems. Conference: 2016 IEEE 34th International Conference on Computer Design (ICCD).

[3] Tanwar, S., Aggarwal, Y., & Dewan, S. (2013, October). Distributed Operating System. International Journal of Research in Information Technology (IJRIT) , 1(10), 50-56.

[4] Mullender, S.J. (1987, September). Process Management in a Distributed Operating System. Proceedings of the International Workshop on Experiences with Distributed Systems, 38-51.

[5] Remzi Arpaci, D., & Dusseau, A.A. (2013, August). Operating Systems: Three Easy Pieces. Lulu Press.

[6] Kai, H., Dongarra, J., & C. Fox, G. (2011, October). Distributed and Cloud Computing: From Parallel Processing to the Internet of Things. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

[7] Buckl, C., Knoll, A., & Schrott, G. (2006, November). Model-Based Development of Fault-Tolerant Embedded Software. Leveraging Applications of Formal Methods, Verification and Validation, 2006. ISoLA 2006. Second International Symposium

[8] Pradeep K, S. (1996, December). Distributed Operating Systems: Concepts and Design (1 ed.). Wiley-IEEE Press.

[9] Abraham, S., Galvin, P.B., & Gagne, G. (2008, July). Operating System Concepts. Wiley Publishing.

[10] Tanenbaum, A.S., & Mullender, S.J. (1981, July). An overview of the Amoeba distributed operating system. ACM SIGOPS Operating Systems Review, 15(3), 51-64.

[11] Yazici AM. BIG DATA AND AMOEBA ORGANIZATIONS: INNOVATIVE MANAGEMENT IN DECISION MAKING. R&S-Research Studies Anatolia Journal. 2024;7(4):519-51.

[12] CHOWDHURY MA, KHONDAKER RM. Amoeba Management System in Japanese Companies: With Special Focus on Inamori Model at Kyocera Corporation. 南山経営研究. 2024 Oct 30;39(1-2):1-26.

[13] Grigoriadou S. Public Budgeting: An Amoeba of Public Policies. InThe Role of the Public Sector in

Essential Components. International Journal of Information Engineering and Electronic Business. 2024 Dec 8;16(6):10-5815.

Building Social and Economic Resilience: A Public Finance Approach 2024 Nov 16 (pp. 119-131). Cham: Springer Nature Switzerland.

[14] Rana MR, Baul S. An Overview of Operating Systems Based on Microkernel Technology and their