فهرست مطالب

بررسی قدرت تحمل نقص (تولرانس نقص) در شبکه محاسباتی
و اولین گام به سوی تلرانس نقص سیستم چندعاملی

چکیده

شبکه محاسباتی بر اساس پایه‌های سخت‌افزاری و نرم‌افزاری‌ای تعریف شده است که منابع به اشتراک گذاشته شده را بر اساس سازماندهی دینامیک و پویا، هماهنگ می‌کند. در شبکه محاسباتی، احتمال وجود نقص خیلی بیشتر از محاسبات موازی قدیمی است. بنابراین، تولرانس نقص یکی از مهمترین ویژگی‌ها در دستیابی به اطمینان و در دسترس بودن و کیفیت خدمات است. در این مقاله، ما بررسی‌ای بر روی تکنیک‌های مختلف تولرانس نقص، مدیریت نقص در سیستم‌های مختلف و موضوعات مرتبط، ارائه می‌دهیم. بررسی سرویس تولرانس نقص با انواع مختلفی از منابع نقص، که شامل نقص پروسه، نقص پردازنده و نقص‌های شبکه است. این مطالعه رابطه‌ی بین نتایج تحقیق در مورد تولرانس نقص در مناطق کاربردی مشخصی از زیربنای شبکه را ارائه می‌دهد و همچنین دستورالعمل‌هایی در مورد تکنیک‌های تولرانس نقص در آینده به ما ارائه می‌دهد که مرجع مناسبی برای محققان است.

وجود اخطارهای خطا و نواقص در سیستم‌ها، هنوز مشکلی است که نیاز به بررسی دارد. این مقاله سعی کرده که مسائلی این چنینی را در سیستم‌های چندعاملی بررسی کند. برای حل آن ما چگونگی استفاده از دو مکانیزم را تجزیه و تحلیل می‌کنیم: خودشفادهی و خودسازماندهی، تا اینکه نقص موجود در سطح عامل را بخوبی کل سطح زیربنایی شبکه، مدیریت کنیم. برای درک بهتر این موارد ما استفاده از عوامل تطبیقی که تغییر موقعیت آنها

بر طبق موقعیت جاری است و تصمیماتشان را برای دیگر عوامل بوسیله پروتکل ارتباطات شایع انتقال می‌دهند را پیشنهاد می‌کنیم.

واژگان کلیدی: تولرانس نقص، قابلیت اطمینان، بازیابی نقطه مقابل، افزونگی، زمان‌بندی، ذخیره‌سازی پیشرفته، گردش کار، سرویس توافق‌نامه سطح (SLA)، عامل‌گرا، بالانس بار و تخصیص.سیستم چند عاملی، خودشفادهی، خودسازماندهی، ارتباطات شایع

بخش اول : بررسي قدرت تحمل نقص (تولرانس نقص) در شبکه محاسباتي

۱. مقدمه

همانطور که عرصه و استفاده‌های سیستم‌های چند عاملی در حال افزایش دست، همچنین برای سیستم‌ها لازم است که هر ۲۴ ساعت بطور الزامی بالا آمده و کار کنند. اگر کوچکترین اشتباه یا خطایی رخ دهد، لازم است که در این زمان خرابی به حداقل برسد.

سیستم‌های موجود در شرکت‌های بزرگ امروزه نمی‌توانند نواقصی را تحمل کند که بر موجودیت سیستم تأثیر می‌گذارد. متأسفانه چیزهای بدی رخ می‌دهند و این چیزها ممکن نیست در سیستم‌های ساده اتفاق بیافتند. موارد گزارش شده از نقص توسط شرکت‌های بزرگی همچون Skype در سال ۲۰۰۷، نشان داد که این سیستم‌ها بی‌نقص نیستند، هرچند که می‌خواستیم باور کنیم که بی‌نقص هستند. در مورد چنین نقصی، ما چکار می‌توانیم انجام دهیم تا آن را به سطح قابل قبولی کاهش دهیم یا حتی آن را بازیابی کنیم. بنابراین برای انجام اینکه ما چه چیزی لازم داریم، این چیزی است که سیستم تلرانس نقص را می‌سازد.

شبکه محاسباتی شکلی از محاسبات تعمیم یافته است که درگیر هماهنگی و به اشتراک‌گذاری توان محاسباتی، داده، و ذخیره سازی و دسترسی پویا (دینامیک) به منابع شبکه و سازماندهی اطلاعات توزیع شده از لحاظ جغرافیایی است [۱]. مدیریت این منابع پیچیده‌تر از منابعی است که به از لحاظ جغرافیایی توزیع شده‌اند و با هم ناهماهنگ بوده، توسط اشخاص یا سازمان‌های مختلفی و با شیوه‌های مخصوص خودشان تولید شده‌اند، و دارای دسترسی‌های متفاوتی و بارهای مختلف دینامیکی و دسترس‌پذیری، هستند.

برای دستیابی به پتانسیل‌های محتمل در شبکه‌های محاسباتی، پس از منابعی که از لحاظ جغرافیایی توزیع شده‌اند، تولرانس نقص اساساً مهم است. علاوه بر این احتمال نقص خیلی بیشتر از محاسبه موازی به روش سنتی است و نقص منابع بر روی برنامه‌های کاری اثر مخربی دارد. و این دلیل تحقیق در مورد تکنیک‌های تولرانس نقص در شبکه محاسباتی است. تولرانس نقص، توانایی یک سیستم در اجرای صحیح عملیات حتی در هنگام وقوع نقص است و این سیستم را بیشتر قابل اطمینان می‌سازد. تولرانس نقص، نشانه بقا در سیستم کامپیوتری است: تشریح کاربرد تولرانس نقص «حفظ تحویل سرویس‌های موردانتظار با وجود رخ دادند نقص در درون خود سیستم، خطاها پیدا شده و درست می‌شوند، و نواقص دائمی مشخص شده و حذف می‌شوند تا زمانی که سیستم به سرویس مورد انتظار تحویل خود ادامه می‌دهد.» سرویس تولرانس نقص برای قانع ارائه‌ی سرویس کیفیت لازم، در شبکه محاسباتی ضروی است و بستگی به انواع مختلف خطاهای مرجع دارد که شامل خطای پروسه، خطای پردازشگر و خطاهای شبکه است. ممکن است که به سمت خطاهای منبع/ کاربرد، سرویس توافق نامه سطح (SLA) و تنزل دادن انتظارات کاربر از سرویس خدمات، سوق یابد.

۱-۱. خلاصه‌اي از تلرانس نقص

تلرانس نقص ویژگی‌ای است که به سیستم اجازه می‌دهد که با وجود نقص در برخی از اجزاء آن، به عملیات خود ادامه دهد. این رویه تنها مربوط به عامل‌های مستقل نیست، بلکه می‌تواند برای عوامل برهم‌کنشی در سیستم چندعاملی نیز کاربرد داشته باشد.

هنگام کارکردن بر روی چنین سیستمی خوب است که متعهد شویم که ما می‌دانیم که چه چیزی مورد تهدید قرار گرفته و رفتار صحیح کدام است و چه چیزی برای اینکه بتوان نقص را پیدا کرد، مناسب است. این مسئله در

خیلی از مقالات مورد تجزیه و تحلیل قرار گرفته است. یکی از بیشترین استفاده‌های نواقص طبقه‌بندی شده در [۷] شرح داده شده است، برای دسته‌بندی نواقص از هفت ویژگی استفاده شده است: فاز مرحله ایجاد شدن یا رخ دادن، مرزهای سیتم، ابعاد، دلایل پدیده‌شناسی، هدف، توانایی، پایداری.

برطبق [۹] این مدل نیازمند توسعه سیستم چند عاملی توسط «مرحله جدیدی از ایجاد و وقوع مرحله‌ای به نام، مرحله خودگردانی» است.

۲-۱. خلاصه‌اي از سيستم چند عاملي

سیستم چند عاملی (MAS)، شبکه ضعیف بهم‌پیوسته‌ای از عوامل نرم‌افزاری است که بر روی حل مسائلی که در ظرفیت های مستقل یا آگاهی از چگونگی حل هر مسئله، کار می‌کند.

با تجزیه و تحلیل ابزارهای پیشرفته‌ی عرصه سیستم چند عاملی می‌توانیم دریابیم که در این ابزارهای زیادی وجود دارند که خیلی بهتر ازارهای مسائلی است که شما می‌خواستید آنها را حل کنید و دانش خود را توسعه دهید. برخی از این ابزارها عبارتند از: Retsine [۱۰]، AgentBuilder، Jake [۱۷]، Jade [۲].

مقایسه‌ی بسط یافته از این پایگاه‌ها را می‌توان در مقاله [۱۳] یافت. ملاک اصلی ما مبنی بر انتخاب Jade به عنوان محیطی برای توسه سیستم چند عاملی است، این است که: آشنایی با محیط خاص، نوع مجوز، توانایی سازگاری سیستم، ظرفیت سیستم برای حمایت از برهم‌کنش پیشرفته، ابزارهای اشکال‌زدایی و...

۲. بررسي خودشفادهي و خودسازماندهي

۲-۱. خلاصه‌اي از خود

هنگامی که در مورد سیستم‌های تلرانس نقص در محیط چند عاملی صحبت می‌کنیم، منظور ما سیستم معقولی است که قادر به درمان کردن و بازیابی نواقص باشد. همچون سیستمی در ادبیات موجود است که در دوره سیستم خودمدیریتی تعریف شده است.

با گسترش این توانایی و فعالیت است که سیستمی قادر به رویارویی و مدیریت خود به سمت دسته‌بندی به عنوان چنین سیستم‌های برطبق خصیصه‌هایی است که هر دسته بر طبق آن طرح‌ریزی شده است.

بنابراین ما می‌توانیم داشته باشیم: خودشفادهی، خودسازماندهی، خودبهینه‌سازی، خودحفاظتی و...

در این لحظه ما بر روی ویژگی‌های خودشفادهی در سطح عامل و خودسازماندهی برای کل مهندسی سیستم متمرکز می‌شویم.

۲-۲. خودشفادهي

خودشفادهی رویکرد مدرنی از مسائل برای یافتن نقص، تشخیص و بازیابی سیستم جهت قابل‌اطمینان‌تر ساختن سیستم است. تجزیه و تحلیل ما بر تعمیر و بازیابی سیستم متمرکز شده است.

به اولین چیزی که ما در مورد مکانیسم خودشفادهی فکر می‌کنیم، این است که ما به چه چیز برای تعیین موقعیت‌های سیستم نیاز داریم. معمولاً همانطور که در مقاله [۱] نشان داده شده است، سه موقعیت اصلی در هر سیستم وجود دارند که می‌توانند: موضعیت نرمال، موقعیت تنزل یافته، موقعیت منقطع شده باشد. برای این منظور خیلی مهم است که برای هر سیستمی مشخص شود که چه ویژگی‌های برای این مکانیسم موردنظر نیاز است.

در اینجا رویکردهای زیادی درخصوص چگونگی یک سیستم خودشفادهی وجود دارد [۶] و [۱۴] اما همه این رویکردها حول یک مکانیسم اصلی می‌گردند: یافتن، تشخیص دادن، تجزیه و تحلیل کردند، برنامه‌ریزی، آگاهی، بازیابی.

یافتن نقص در سیستم‌هایی که دارای مکانیسم خودشفادهی هستند معمولاً بوسیله مکانیسم ایی که قادر به شناسایی موارد خراب شده باشد، بدست می‌آید. برای درک این موضوع، مقالات مختلفی دو رویکرد اساسی برای عیب‌یابی و گزارش رفتار مناسب، ارائه داده‌اند: جستجو برای ناسازگاری‌های موجود در ارسال داده یا رها کردن رفتار نامناسب [۴].

رویه‌ها (روش‌ها) برای سیستم‌های خودشفادهنده خیلی مهم هستند. در [۴] ما سه گروه از این رویه‌ها را تعریف کردیم: رویه‌های عمل، رویه‌های هدف، رویه‌های کاربردی مفید.

و در پایان تکنیک‌های بازیابی که قابل بکارگیری هستند را نشان می‌دهیم: جایگزینی، تراز (متعادل کردن)، جداسازی، ثبات، تعیین جهت مجدد، تعیین موقعیت مجد، گوناگونی.

۳-۲. خودسازماندهي

تعریف‌های زیادی از خودسازماندهی در خیلی از زمینه‌های تکنیکی و عرصه‌های اجتماعی، ارائه شده است. از نقطه نظر علوم کامپیوتر، این روند بیشتر به سمت درک طبیعت چیزها، تمرکز دارد زیرا معمولاً این آگاهی موجب عملکرد بهتری می‌شود. بر طبق [۳] «خودسازماندهی همچون مکانیسمی یا فرایندی که سیستم را قادر به تغییر سازماندهی بدون هیچ دستور خارجی در طول زمان اجرا می‌سازد، تعریف شده است.» در این مقاله نویسنده نیز خودسازماندهی را همچون ضعف یا قوتی بر طبق چگونگی حضور یا کنترل مرکزی و عامل برنامه‌ریزی شده، دسته‌بندی کرده است. گونه‌ای از سیستم خودسازماندهی نیاز به قابلیت تطابق، تکامل و ظهور دارد.

هنگامی که مکانیسم‌های یک خودسازماندهی‌ای ایجاد می‌شوند، برخی موارد دیگری نیز وجود دارد که باید به حساب آورده شده برای سیستم درنظر گرفته شوند [۵]: تعیین مرزها بطور اتوماتیک، عملیات بسته و آزادی متحرک، استقلال هویت و ساختار، تعمیر و نگهداری (خودشفادهی)، فیدبک، فوریت، ضروریت، عکس‌العمل اتوماتیک برای اختلالات، کاهش پیچیدگی.

برطبق [۳] این رویکردها، هنگامی که سیستم خودسازماندهی طراحی می‌شود، می‌تواند مورد بررسی قرار گیرند و بسته به مکانیسم‌هایی که بر اساس آن طراحی شده‌اند می‌توانند به پنج دسته تقسیم شوند: برهم‌کنش‌های مستقیم مابین عواملی که از اصول اساسی استفاده می‌کند مانند انتشار و موضع‌یابی، برهم‌کنش‌های غیرمستقیم مابین عوامل، تقویت رفتارهای عامل، همکاری رفتارهای عوامل مستقل، انتخاب معماری اصلی.

یکی از مسائل اساسی در سیستم چندعاملی، توافق مابین عوامل است. برای حل این مسئله گام بعدی بکارگیری یک پروتکل ارتباطی خوب است. چنین مکانیسمی پروتکل مبنی بر شایعه است که برای سنجش اینکه آیا قادر به حفظ اتصال در مقایس بزرگ سیستم‌های پویا با وجود تکان‌های شدید و دیگر نواقص، آزموده شده است.

مثال‌هایی از پروتکل‌های مبنی بر شایعه خبرپراکنی است [۸]، T-MAN [۱۲]، پروتکل رتبه‌بندی، پروتکل Silver [۱۵] و غیره.

۳- شرح پروژه

در این مقاله ما معماری کلی و اجزاء یک سیستم چندعاملی که با استفاده از Jade ساخته شده است را نشان می‌دهیم [۲].

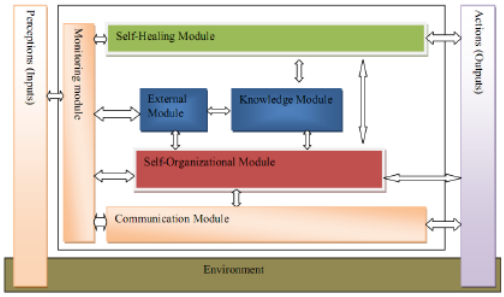
۳-۱. معماري سيستم

مدل ما شامل شبکه‌ای از عوامل است که برای هرکدام مأموریت خاصی تعریف شده است و این ارتباط برای تبادل پیام بکار گرفته می‌شود. این سیستم درصدد این است که بتواند قادر به برقراری ارتباط و همچنین با دیگر عواملی نیز که در محیط خارج از محدوده‌ی موردنظر ماست، باشد. در اینجا رویکردهای دردسترس متعددی برای برای ایجاد و طراحی یک سیستم تلرانس نقص در سیستم‌های چند عاملی وجود دارد. ما چهارچوبی بر اساس روشی پیشنهاد می‌کنیم که برای هر عامل بیشترین ابزار پیشرفته چند عاملی در سطح تلرانس نقص را ارائه دهد. این سطح در زیربخش بعدی نشان داده خواهد شد.

۳-۲. معماري عوامل

عوامل پیشنهادی ما درصدد این هستند که بتوان عوامل را قادر به کنترل خودکار تصمیماتشان در موقعیت‌های مختلف، ساخت. آنها با شرایط سیستم وفق داده خواهند شد و تصمیمات بر اساس وضعیت داخلی و اطلاعات محیطی گرفته خواهد شد. این تصمیمات بهترین راه‌حل‌ها را برای رفاه در سراسر سیستم، ارائه خواهند داد. عوامل به سمت عوامل مشارکتی با مشخصات عمومی زیر سوق خواهند یافت: مهارت‌ها (چیزی که قادر به انجام آن هستند)، استعداد اجتماعی (همکاری)، دانش (در مورد خود، دیگر عوامل، محیط) و غیره.

معماری ارائه شده در سطح عامل:



شکل ۱. معماری سطح تلرانس نقص عامل

۳-۳. ماژول‌هاي عامل

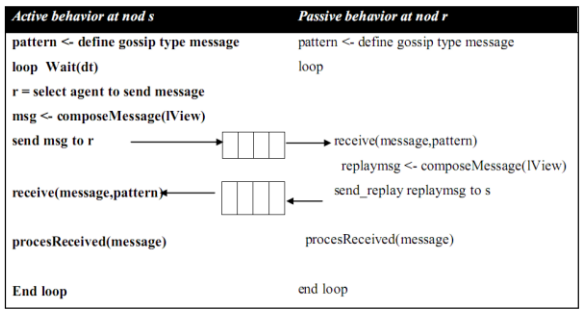
همانطور که در شکل بالا می‌بینیم، راه‌حل ما مبنی بر شش ماژول در لحظه است: نظارت، ارتباطات، دانش، خارجی، ماژول سازماندهی و شفادهی. شمار این ماژول‌ها می‌تواند در مواقعی افزایش یابد.

۱-۳-۳. ماژول نظارت

این ماژول به سمت مکانیسم از ماژول‌های خودشفادهنده مابین محیط و سازمان، تمایل دارد. همراه با ماژول‌های ارتباطی، گرایش به راه‌حلی کامل برای تمام اطلاعات دریافتی و تحویل داده شده از محیط، دارد.

۲-۳-۳. ماژول ارتباطي

این ماژول مسئول برقراری ارتباط مابین عوامل است. که یکی از ماژول‌های خیلی مهم است چرا که یک ماژول خودسازمان دهنده است که می‌تواند این کار را نیز انجام دهد. پروتکل بسط یافته‌ی ما از پروتکل اصلی تبعیت می‌کند:



شکل ۲. پروتکل ارتباطی

۳-۳-۳. ماژول دانش

ما درصدد استفاده از این ماژول برای ذخیره‌سازی تمام اطلاعات لازم برای دیگر ماژول‌ها هستیم. که دیگر ماژول‌ها در ارتباط خواهد بود و دیگر ماژول‌ها را قادر به ذخیره‌سازی اطلاعات مربوطه در طول پردازش و پس از آن، می‌سازد.

هر ماژول متصل به معماری‌ای است که در مربوط به ارتباطات آسان با ماژول دانش است. که این ماژول مسیرهای نرمال و غیرنرمال و همچنین فعالیت‌هایی که نیازمند جایگری در موارد راه‌حل‌های خاصی که قبلاص توسط دیگر عوامل عیب‌یابی شده‌اند را ذخیره می‌کند و به دست یافتن به سیستم تلرانس نقص کمک خواهد کرد.

۳-۳-٤. ماژول خارجي

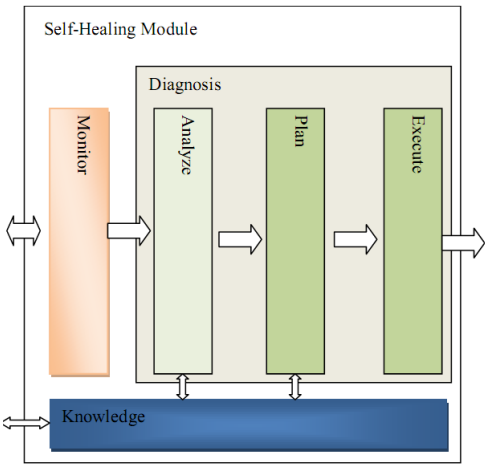
ماژول خارجی برای استفاده از اطلاعات جمع‌آوری شده و نگهداری از پایگاه داد نقص‌های شناخته شده و راه‌حل‌های آنها و همچنین برای تعامل با محیط خارجی است.

۳-۳-٥. ماژول خودشفادهنده

این ماژول مسئول یافتن و بازیابی نقص‌های سیستم است.

ساختار ماژول:

- مانیتور- واسطی مابین این ماژول و یک مانیتور ارائه می‌دهد. که نیازمندیها را پالایش کرده و آنها را به صورت داده‌های قابل ارائه تغییر می‌دهد که توسط موتور تشخیص راحت‌تر قابل پردازش باشد.
- ارتباطات- منبی بر ارسال اطلاعات درخصوص موقعیت آن در سیستم.
- دانش- که این واسطی برای ماژول دانش مرکزی است.
- تشخیص- تجزیه و تحلیل (مسئول تجزیه و تحلیل موقعیت)، طرح‌ریزی (بر اساس عملیات موقعیت مناسب ایجاد و تجزیه و تحلیل شده‌اند)، اجرا (عملیات ارائه شده در مرحله طرح‌ریزی که اجرایی هستند).



شکل ۳. ماژول خودشفادهنده

۳-۳-۶. ماژول سازمان‌دهنده

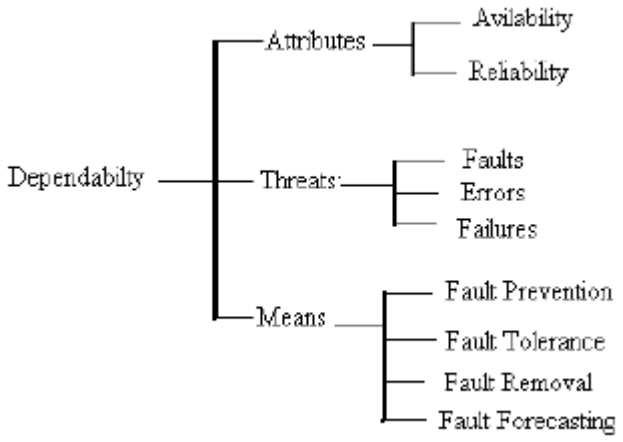
ماژولی است که از الگوریتم خودسازماندهنده برای معماری پیشنهادی، استفاده می‌کند. این ماژول مسئول انتخاب عواملی که است که عامل موجود نیاز به برقراری ارتباط با آن برای دستیابی مطلوب است. این ماژول با کاربرد دیگر ماژول‌ها درارتباط است: واسط ارتباطات (ارائه واسط مابین ماژول رایج و ارتباط با آن)، واسط دانش (واسطی با ماژول دانش مرکزی است).

جدا از این واسط‌ها ما همچنین دارای ماژول‌های تشخیص موقعیت و ماژول مدیریت موقعیت هستیم که درخصوص اطلاعات لازم برای ذخیره‌سازی یا ارسال تصمیم می‌گیرند.

بخش دوم : اولين گام به سوي تلرانس نقص سيستم چندعاملي

۱. طبقه‌بندي قابليت اطمينان

تولرانس نقص سبب دسترسی به قابلیت اطمینان سیستم می‌شود. قابلیت اطمینان وابسته به برخی جنبه‌های QOS (وضعیت کارایی یک شبکه- کیفیت خدمات) ارائه شده توسط سیستم، شامل برخی ویژگی‌ها از قبیل قابلیت اطمینان و در دسترس بودن، است. طبقه‌بندی اطمینان در شکل ۱ نشان داده شده است.



شکل ۱. طبقه‌بندی قابلیت اطمینان

قابلیت اطمینان نشان‌دهنده‌ی این است که سیستم قادر به اجرای مداوم بدون خرابی است. یک سیستم با قابلیطت اطمینان بالا، سیستمی است که به طور مداوم بدون هیچ گونه قطع شدنی در طول یک زمان مداوم نسبتاً طولانی، کار کند. دسترس‌پذیری بدین معنی است که سیستم بلافاصله برای استفاده کردن، آماده باشد. تکنیک‌های تلرانس عیب، اغلب برای افزایش دسترس‌پذیری و قابلیت اطمینان، استفاده می‌شوند.

قابلیت اطمینان $R(t)$: احتمال احتمال استفاده از سیستم بطور مداوم در یک زمان مشخص است $[0,t]$. قابلیطت اطمینان دارای رابطه‌ی نزدیکی با زمان متوسط خرابی (MTTF) و زمان متوسط مابین خرابی‌ها (MTBF) دارد. MTTF میانگین زمانی است که سیستم کار می‌کند تا اینکه خرابی رخ می‌دهد، درحالیکه MTBF

میانگین زمان مابین دو خرابی متوالی است. تفاوت میان این دو ناشی از زمانی است که سیستم نیاز به تعمیر خرابی اول دارد. زمان لازم برای تعمیر بوسیله MTTR نشان داده می‌شود، که ما آن را از رابطه MTBF=MTTF+MTTR بدست می‌آوریم.

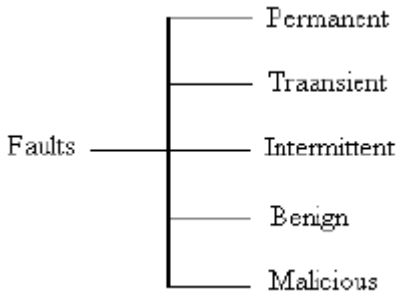
دسترس‌پذیری A(t): میانگین کسر زمان در طول فواصل زمانی است که سیستم در حال کار کردن است [0,t]. این دسترس‌پذیری می‌تواند از MTTF، MTBF و MTTR بصورت زیر، بدست آید:

A=MTTF/MTBF=MTTF/(MTTF+MTTR)

و این برای سیستمی با قابلیت اطمینان پایین جهت داشتن سیستمی با قابلیت اطمینان بالا، امکان‌پذیر است: در خصوص سیستمی که به طور میانگین در هر ساعت دچار نقص و خرابی شود اما فقط پس از دو ثانیه پشتیبان از آن تهیه شود. همچون سیستمی که دارای MTBFی در هر یک ساعت و درنتیجه، دارای قابلیت اطمینان پایین باشد؛ اگرچه دسترس‌پذیری آن بالا است: A = ۳۵۹۹/۳۶۰۰ = ۰/۹۹۹۷۲

تاکنون، روش‌های پیشرفته‌ای برای تأمین قابلیت اطمینان منابع شبکه بوجود آمده‌اند که به طور متوسط دارای روش‌های پیشرفته‌ای جهت تلرانس نقص هستند. تلرانس نقص شامل: ۱) یافتن نقص‌ها و خرابی‌ها در منابع شبکه و ۲) بازیابی جهت اجازه‌ی ادامه‌ی محاسبات.

اخطارهای خطا و خرابی‌ها، به عنوان نواقص طبقه‌بندی می‌شوند. نقص (یا خرابی)ای می‌تواند خرابی سخت‌افزاری یا نرم‌افزاری/ خطای برنامه‌ای (اشکال)، باشد. یک اخطار خطا جلوه‌ای از نقص/ خرابی/ اشکال است. نقص سخت‌افزاری می‌تواند بصورت عنوان دائمی، گذرا، متناوب، بی‌خطر یا پرخطر، دسته‌بندی شود (شکل ۲).



شکل ۲. طبقه‌بندی نقص

خطای دائمی منجر به دائمی شدن در اجرای اجزاء می‌شود. نقص گذرا، نقصی است که سبب تأثیر بر اجزا در برخی مواقع می‌شود؛ که پس از مدت زمانی رفع می‌شود و کارکرد اجزا به طور کامل بازیابی می‌شود. خطای متناوب هرگز کاملاً رفع نخواهد شد؛ مابین ساکن بودن و فعال بودن در حال نوسان است. هنگامی که نقص سکن است، اجزا کاربردی نرمال هستند، هنگامی که نقص فعال می‌شود، این اجزا بد عمل می کنند. نقصی که فقط سبب ساکن مانند فقط یک بخش می شوند، بی‌خطر نامیده می‌شود. ماند نواقصی که در تبادلات وجود دارند. نواقص پرخطر آنهایی هستند که موجب می شوند یک واحد تولیدی (ورودی) معقول به نظر برسد، اما در واقع، خروجی است، یا اجزاء را می‌سازد: عملی بدخواهانه، و مقادیر متفاوت خروجی برای دریافت‌کننده‌های متفاوتی می‌فرستند.

برای یک عامل جهت‌دار، تلرانس نقص بازدارنده در چارچوب شبکه برای نواقصی بکار گرفته شده است که به شش دسته تقسیم شده‌اند: a) نواقص سخت‌افزاری: CPU، حافظه، ذخیره‌سازی، b) نواقص کاربردی و سیستم عاملی: نفوذ در حافظه، در دسترس نبودن منابع، c) نواقص شبکه: خرابی گره، خرابی پیوند، گم شدن بسته، d) نقص نرم‌افزاری: عدم استفاده از استثناء، ورودی غیرمنتظره، e) نواقص منابع: نواقص حجم و f) نواقص خارج از زمان موردنظر. اینها به دسته‌های دیگری نیز تقسیم شده‌اند، هنگامی که این عوامل در حال تبادل با نواقص

بازدارنده هستند. عوامل ذکر شده دارای اطلاعاتی درخصوص شرایط سخت افزار، مصرف حافظه برای پروسه‌های در حال اجرا، منابع در دسترس، شرایط شبکه و اجزا زمان متوسط نقص هستند. بر اساس این اطلاعات و شرایط بحرانی، این عوامل سیستم شبکه را قادر به تلرانس نقص می‌کنند [۵].

نشان داده شد که حذف، برهم‌کنش و زمان نواقص، در محاسبات شبکه خیلی رایج هستند. نواقص حذفی هنگامی که منابع در دسترس نیستند، افزایش می‌یابد. برهم‌کنش ممکن است ناشی از سپاورت پروتکل‌های مختلف توسط سرویس‌های مختلف، ناسازگاری امنیتی و مشکلات رویه‌ای، باشد. نواقص زمانی، هنگامی افزایش می‌یابند که سرویس ممکن است سرویس دیگری را به دلیل خارج از موعد بودن، بلوکه کرده باشد.

این مقابل بصورت زیر تنظیم شده است. بخش دو خلاصه‌ای از سرویس‌های مختلف استفاده شده برای یافتن خرابی‌ها در هر نوع منابع شبکه‌ای، ارائه می‌دهد؛ بخش سه، استراتژی‌های تلرانس نقص، مکانیسم‌ها، تکنیک‌هایی شبیه نقطه مقابل و تکرار را شرح می‌دهد. بخش چهار، با شبکه‌های تلرانس نقص سروکار دارد، بخش پنج، چگونگی مدیریت نقص صورت گرفته در مجاورت مناقط کاربردی و زیربنای شبکه را ارائه می‌دهد و بخش شش یافته‌ها و نتایج را نشان می‌دهد.

۲. سرویس‌هاي يافتن خرابي

تا آنجا که یافتن نقص در هر گونه منبعی از شبکه حائز اهمیت است، در این جا دو سرویس اصلی وجود دارد: مدل کشیدن و مدل هل دادن، همانطور که شرح داده شده است [۱۰]. در مدل کشیدن، اجزا مختلف شبکه برای ارسال متناوب سیگنال به یابنده نقص، مسئول هستند. در صورت فقدان هر گونه سیگنالی از هر کدام از اجزاء شبکه، یابنده نقص، این نواقصی را که در اجزا شبکه رخ داده است، شناسایی می‌کند. سپس ابزارهای مناسب

توسط مکانیسم‌های تلرانس نقص از پیش تعریف شده را بکار می‌گیرد. در مدل هل دادن، اجزاء یابنده نقص مسئول ارسال سیگنال‌های متناوب به اجزاء مختف شبکه هستند. بعلاوه، اجزا یابنده نقص مسئول یافتن نواقص مختلف از قبیل نواقص شبکه، خرابی‌های گره، نواقص پروسه‌ها و نواقص پردازشگر هستند.

۳. استراتژي تلرانس نقص

تلرانس نقص می‌تواند با استراتژی زیر بدست آید: پوشش نقص، پروسه‌ای از نواقص بازدارنده در سیستم از خطاهای شناسایی در سیستم است. این اساساً برای پنهان کردن نواقص رخ داده و نواق و نواقص رایج در نتایج اخطار خطا، استفاده می‌شود. پوشش نقص، در سیستم‌هایی که نمی‌توانند حتی اجازه ایجاد نتایج خطاهای آنی را بدهند، استفاده می‌شود. برای مثال: اخطار خطای حافظه صحیح و اکثریت آراء. پیکربندی دوباره، فرایند جهت حذف اجـــزا نقص از سیستم و بازیابی سیستم از برخی حالت‌های عملیاتی است. پیکربندی‌دوباره به شرح زیر: ۱) یافتن نقص فرایندی از شناسایی نواقصی است که رخ داده‌اند. یافتن نقص، اغلب قبل از هر گونه روش بازیابی‌ای که می‌تواند وجود داشته باشد، لازم است. ۲) موقعیت نقص فرایندی از تعیین جایی که نقص رخ داده‌است که برای بازیابی مناسب لازم است. ۳) محدود کردن نقص، فرایندی از منزوی کردن نقص و جلوگیری از تأثیر نقص بر روی انتشار آن در کل سیستم است. ۴) بازیابی نقص فرآیندی از بازیافتن موقعیت‌های عملیاتی یا عوامل باقیمانده بوسیله پیکربندی دوباره حتی با وجود نواقص است.

۳-۱. مکانیسم‌ها

مکانیسم تلرانس نقص در محیط شبکه می‌تواند به دو گونه اصلی تقسیم شود: پیش فعال و پس‌فعال. در پیش‌فعال، مکانیسم تلرانس نقص به خرابی موجود در شبکه رسیدگی می‌کند که این کار قبل از برنامه‌ریزی برای عمل انجام می‌شود، و به امید این که این عمل با شکست مواجه نشود، فرستاده می‌شود.

۳-۲. تکنیک‌ها

در سیستم‌های توزیع شده به طور کلی، روش‌های بازیابی در سیستم‌های شبکه مبتنی بر بهره‌برداری از افزونگی است. افزونگی، کلید تلرانس نقص است. می‌تواند هیچگونه FTای بدون افزونگی وجود داشته باشد! دو نوع افزونگی وجود دارد: افزونگی موقتی و افزونگی فاصله‌ای (فضایی). افزونگی موقتی درگیر تلاش‌های مکرر برای راه‌اندازی مجدد منابع و سرویس‌های خراب شده است. افزونگی فاصله‌ای برای سود جستن از کپی‌برداری‌های چندگانه از منابع محاسباتی است.

تلرانس نقص در سیستم کامپیوتری بوسیله افزونگی در سخت‌افزار، نرم‌افزار، اطلاعات، و/یا محاسبات، بدست می‌آید. این گونه افزونگی می‌تواند در پیکربندی‌های آماری، دینامیکی و یا هیبریدی صورت پذیرد. افزونگی، تلفیق اجزا اضافی در طراحی سیستم است و بنابراین کارکرد آن قابل خراب شدن حتی در طول یک نقص (خرابی)، نیست. این علاوه بر اطلاعات اضافی، منابع/ زمان لازم برای به عملیات یک سیستم معمولی است.

چهار شکل از افزونگی به قرار زیر است: افزونگی سخت‌افزاری، افزونگی نرم‌افزاری افزونگی اطلاعاتی و افزونگی زمانی. نقص‌های سخت‌افزاری معمولاً با استفاده از افزونگی سخت‌افزاری، اطلاعاتی یا زمانی سروکار دارند، درحالیکه نقص‌های نرم افزاری در مقابل، بوسیله افزونگی نرم‌افزاری، محافظت می‌شوند.

افزونگی سخت‌افزاری: توسط سخت‌افزارهای اضافی تلفیق شده در طراحی برای هرگونه نقص یا باطل کردن تأثیرات محاسبات اشتباه، تهیه شده است. سخت‌افزار اضافی، برای باطل کردن اثر محاسبات اشتباه، اضافه شده است. برای مثال، به جای داشتن یک پردازنده، می‌توانیم از دو یا سه پرازنده استفاده کنیم، که هر کدام کاربرد مشابهی دارند. با داشتن دو پردازنده، می‌توانیم خرابی‌های یک پرازنده را عیب‌یابی و رفع کنیم؛ با داشتن سه پردازنده، می‌توانیم از اکثریت خروجی‌ها برای باطل کردن خروجی‌های اشتباه در یک پردازنده معیوب، استفاده کنیم. این مثالی برای افزونگی سخت‌افزاری آماری است که بطور عینی و بلافاصله خرابی را می‌پوشاند. شکل مختلفی از افزونگی سخت‌افزاری، افزونگی پویا است، هنگامی که، اجزاء ذخیره به محض رخ دادن نقص فعال شده و جزئی از اجزا فعال جاری شوند. ترکیب تکنیک‌های افزونگی دینامیک و آماری، ممکن است، منجر به افزونگی سخت‌افزاری هیبریدی شود.

افزونگی نرم‌افزاری: این علاوه بر نرم‌افزارهای اضافی، برتر از چیزهایی است که که برای اجرای یک عملیات، عیب‌یابی و تلرانس نقص ممکن، لازم است. در محصولات مستقل دو یا بیشتر از دو نسخه از نرم‌افزار (بطور مهتر توسط تیم‌های گسسته برنامه‌ریز) وجود دارد به این امید که دو گونه متفاوت بر روی یک ورودی مشابه دچار خرابی نشوند. این گونه از طراحی‌های متفاوت اطمینان می‌دهد که تمام نسخه‌ها بر روی داده ورودی مشابه، دچار نقص نشوند.

افزونگی اطلاعاتی: این علوه بر اطلاعات اضافی برای اجرای کارکرد مشخص شده است. در اینجا، بیت‌های اضافه (بیت‌های مقابله‌ای نامیده شده‌اند) به بیت‌های داده اصلی اضافه شده‌اند بنابراین یک اخطار در بیت‌های داده می‌تواند کشف شده یا حتی اصلاح شود. افزونگی اطلاعاتی اغلب نیازمند افزونگی سخت‌افزاری برای پرازش بیت‌های مقابله‌ای اضافه شده است.

افزونگی زمانی: برای اضافه شدن بر زمان اجرای عملیات در سیستم از قبیل یافتن عیب و اغلب تلرانس نقص، قابل دستیابی است. افزونگی زمان بکار انداختن سیستم از طریق اجرای دوباره یک برنامه بر روی یک سخت‌افزار یکسان است.

استفاده از افزونگی می‌تواند توانایی‌های دیگری به یک سیستم ارائه دهد. اما، افزونگی می‌تواند فشار خیلی مهمی بر روی اجرا، اندازه، وزن، مصرف برق و قابلیت اطمینان یک سیستم داشته باشد.

هر دو افزونگی موقتی و فاصله‌ای در شبکه‌ها بکار رفته‌اند. هنوز، به دلیل اینکه سیستم‌های شبکه بطور ذاتی منابع محاسباتی اضافی ارائه می‌دهند، افزونگی فاصله‌ای بر روی تحقیقات تلرانس نقص متمرکز شده است. سه تکنیک مورد تأکید در افزونگی فاصله‌ای وجود دارد: ۱) نقطه مقابله یا ذخیره کردن دوره‌ای در حالت اجرای منابع محاسباتی حتی در منابع خراب شده، که می‌تواند در منابع مختلف دیگری از سر گرفته شده و مورد استفاده قرار گیرد؛ ۲) تکرار (انعکاس) یا نگهداشتن شماره کافی نسخه‌ها، یا کپی‌های فرایند اجرایی موازی در منابع مختلف که حداقل یک نسخه بدست آمده است؛ و ۳) برنامه‌ریزی مجدد یا یافتن منابع مختلف برای اجرای دوباره وظایف با شکست مواجه شده. توجه کنید که تکنیک ۱ و ۲ با عملیاتی سروکار دارند که در طول زمان تکرار شده و بنابراین جزء افزونگی موقتی هستند. تکنیک‌های تلرانس نقص در سیستم‌های شبکه معمولاً بوسیله نقطه مقابله- بازیابی و عمل تکرار د منابع جایگزینی در موارد قطع سیستم، قابل دستیابی هستند.

۱-۲-۳. نقطه مقابله- بازیابي

نقطه مقابله- بازیابی بستگی به MTTR سیستم دارد. در ذخیره‌سازی دوره‌ای، موقعیت کاربرد ذخیره ثابت، معمولاً بر روی هارددیسک است. بعد از خرابی، این عملیات از از آخرین نقطه مقابله بیشتر از نقطه شروع،

راه‌اندازی مجدد می‌شود. در اینجا سه استراتژی نقطه مقابله وجود دارد: نقطه مقابله‌ی هماهنگ، نقطه مقابله‌ی ناهماهنگ و نقطه مقابله‌ی ارتباطات- القاء شده. در نقطه مقابله‌ی هماهنگ، فرایندهای همزمان نقطه مقابله برای اطمینان از اینکه حالت‌های ذیخره‌شده شامل همگی می‌باشد، است، بنابراین رویهمرفته ترکیبی است، محل ذخیره نیز ثابت است. در مقابل، در نقطه مقابله‌ی ناهماهنگ، فرآیند زمانی نقطه مقابله‌ها بطور مستقل در زمان‌های متفاوتی است و برای پیام به شمار نمی‌آیند. نقطه مقابله‌ی ارتباطات- القاء شده، درصدد هماهنگ کردن تنها بخش بحرانی نقطه مقابله‌ای است [۲۴].

برخی از این نتایج اینها هستند که: در چه سطحی (کاربر یا هسته) ما باید نقطه مقابله داشته باشیم: ما طرفدار یا مخالف کدام سطح هستیم؟ چگونه باید فرایند نقطه مقابله برای کاربرد شفاف شود؟ ما چه تعداد نقطه مقابله‌ای باید داشته باشیم؟ در کدام نقاط در طول اجرای برنامه می‌توانیم نقطه‌مقابله داشته باشیم؟ چگونه می‌توانیم رویهمرفته نقطه مقابله‌ای را کاهش دهیم؟ مسئله تعیین شمار بهینه نقطه مقابله‌ای‌ها به عنوان مسئله تعیین نقطه مقابله شناخته شده است و هدف این است که کل زمان اجرای نقطه‌مقابله، نوع ذخیره نقطه مقابله و موقعیت آن، و فراوانی نقطه مقابله‌ای، به حداقل برسد.

۲-۲-۳. تکرار

تکرار بستگی به دسترس‌پذیری موقعیت‌های جایگزین برای اجرای نسخه‌هاست. در تکرار منابع شبکه، منابع شبکه چندگانه همزمان با هم اجرا در محاسبات مساوی و اجرا شده و در حالت مساوی نیز حفظ می‌شوند. هدف از تکرار این است که اطمینان حاصل شود که حداقل یک نسخه وجود دارد که همیشه قادر به تکمیل محاسبات حتی در طول خطاهای دیگر، است. در برخی موارد، یک نسخه ممکن است به عنوان نسخه اولیه و جهت تعامل خارجی طراحی شده باشد، درحالیکه دیگران آن را همچون پشتیبانی می‌پندارند.

اگرچه روش تکرار بطور وسیعی در تکنیک تلرانس نقص مورد استفاده قرار گرفته است، اما شماری از نسخه‌های پشتبانی کاملاً بی‌فایده هستند. شماری از نسخه‌های پشتیبانی بطور جدی همچون پوششی در مقابل شماری از نواقص افزوده شده، در حال افزایش هستند. همانطور که برخی از این افزایش پشتیبانی به عنوان پشتیبان‌هایی مدیریت می‌شوند که خیلی گران هستند. ترکیب این تکنیک‌ها سبب ایجاد این مسئله شده است. همچون تکنیکی موردپسند جهت استفاده از نواقص متعدد، ظاهر شد. اساساً این یک ایده یدکی (ذخیره-جایگزین) برای تلرانس نقص است که نیازمند اندکی دستگاه‌های پشتانی و سپس تکرار مبتنی بر مشی است. این دستگاه‌های پشتیبانی به بنام ترکیبات متناظر برای هماهنگی کردن دستگاه‌ها نامیده شده‌اند. کلاً در ترکیب مبنی بر تکنیک‌ها در طول بازیابی نواقص، خیلی وسیع است. بنابراین این تکنیک اگر احتمال نقص آن کم باشد، قابل پذیرفتن است. دو جنبه از تحقیقات در خصوص تکرار منابع شامل موارد زیر است: ۱) الگوریتم‌هایی برای تعیین بهینه (یا نزدیک به بهینه) تکرار نسخه‌ها برای افزایش تلرانس نقص و ۲) روش‌هایی برای همزمانی حالات نسخه‌ها برای اطمینان حاصل کردن از ثبات و سازگاری آنها.

برخی از نتایج: درجه تکرار، نسخه مورد تقاضا، ثبات و تعداد نسخه‌های که به عنوان واکنشی در تغییر مشحصات سیستم بکار می‌روند (شمار منابع فعال، فراوانی نقص منابع و بار سیستم).

۴. شبکه‌هاي تلرانس نقص

تلرانس نقص در شبکه‌ها بوسیله داشتن مسیر های چندگانه که منبع را به مقصد وصل می‌کند، و/یا واحدهای نازک که قادر به سویچ کردن در واحدهای خراب شده هستند، قابل دستیابی است.

۴-۱. مقیاس‌هاي شبکه

مقیاس‌های ارائه شده، میزان تخریب در قابلیت اطمینان و اجرا در شبکه کامپیوتر با وجود نواقص، را ارائه می‌دهند. قابلیت اطمینان: قابلیت اطمینان شبکه $R(t)$ در زمان t، همچون احتمالی که همه گره‌ها در حال کار کردن و قادر به اتصال با یکدیگر بطور یکپارچه در خلال مدت $[0,t]$ است. مسیر قابلیت اطمینان/ قابلیت اطمینان ترمینال، به عنوان احتمال اینکه مسیر عملیاتتی برای این منبع- مقصد در طول زمان یکپارچه $[0,t]$ است. پهنای باند: میزان حداکثر پیامی که می‌تواند در شبکه جریان یابد. که این معمولاً مانند گره‌ها یا پیوندها خراب در شبکه کاهش می‌یابد. قابلیت اتصال: تعداد مورد انتظار در زمان t از منابع- مقصد که هنوز با وجود فرایند معیوب در حال اتصال هستند.

۴-۲. توپولوژي‌هاي معمولي شبکه و قابلیت ارتجاع آنها

۱. شبکه خط عرضی: در شبکه‌های خط عرضی، نقص هر کدام از جعبه کلیدها مسلماً هم ورودی و هم خروجی با قطع خواهد کرد. افزونگی می‌تواند به عنوان تلرانس نقص خط عرضی، مطرح شود. ما ستر و ستون به جعبه کلیدها اضافه می‌کنیم و اتصالات ورودی و خروجی را افزایش می‌دهیم، بنابراین هر ورودی‌ای می تواند به دو یا دیگر ستونها ارسال کند، و هر خروجی نیز می‌تواند از دو یا چند ستون دریافت کند. اگر هر کدام از جعبه کلیدها دچار مشکل شوند، این سطر و ستون توسط ستون یا سطرهای کنارشان جایگزین شده و سر و ستون ذخیر تعمیر می‌شوند. **۲. شبکه مستطیلی:** شبکه مستطیلی دوبعدی طبق قرارداد، شبکه‌ای است که قادر به تحمل هیچ گونه نقصی در هیچ یک از گره بدون از دست دادن خصوصیت شبکه، نیست. تلرانس نقص شبکه شامل گره‌های ذخیره‌ای است که قادر به سویچ کردن و قرار گرفتن در هر یک از همسایگی‌هایی که خرابی رخ داده است، هستند [۳۳].

۳-۴. مسیریابي تلرانس نقص

هدف از استراتژی مسیریابی تلرانس نقص، بدست آوردن اندازه منابع در مقصد با وجود زیرمجموعه‌ای از نواقص موجود در شبکه، است [۳۳]. این تصور پایه‌ای ساده است: اگر هیچ گونه مسیر کوتاه یا خیلی مناسب به دلیل گره‌های معیوب، در دسترس نبود، مسیریابی مجدد پیامی توسط دیگر مسیرها در مقصد. این مثال‌ها برای مسیریابی تلرانس نقص هایپرکاب (hypercube) تلرانس نقص و مسیریابی بر پایه مبدأ در شبکه است. ایده اولیه از hypercube مسیریابی تلرانس نقص در لیست ابعادی پیام‌هایی که فرستاده می‌شوند و سپس هر یک از آنها طی مسیر می‌کنند، قرار دارد. همچون لبه‌هایی هستند که پیموده شده‌اند، آنها از لیست حذف شده‌اند. اگر، به دلیل نقص پیوند یا گره‌ای، پیوند مناسب در دسترس نباشد، سپس دیگر لبه‌ها در لیست، یا هر یک، برای پیماش انتخاب شوند. اگر هیچ لبه‌ای در دسترس نباشد (پیام رسیده از برخی گره‌ها برای یافتن تمام ابعاد موجود در لیست انجام شده است)، به سمت گره قبلی برگشت داده می‌شوند و مجدد سعی می‌شود. در مسیریابی بر پایه‌ی مبدأ در شبکه مناطق معیوب در طی پیشروی شناسایی می‌شوند.

۵. مدیریت نقص شبکه در مناطق کاربردي مشخص

در سطح منابع تکنیک‌های تلرانس نقص درگیر کاربرد استاندارد تکنیک‌های تلرانس نقص در هر یک یا همه‌ی منابع موجود در سیستم هستند. این ناهماهنگی یا اختصاص نیافتن نوع سیستم موجب افزایش پیچیدگی سیستم می‌شود. نواقص سطح منابع مانند خرابی‌های گره، نواقص پیوندی، رویدادهای از قبیل خاموش شدن غیرمنتظر دستگاه یا از دست دادن پیوند شبکه‌ای واضحاً به عنوان نواقص شناخته شده‌اند.

تلرانس نقص سطح سرویس، از طرف دیگر، با رویه‌های پهنای سیستم بطور مستقیم برای افزایش قابلیت اطمینان در دوره‌های متناوب سرویس‌دهی، مرتبط است. کیفیت خدمات (QOS)، فاکتور کلیدی است. قابلیت اطمینان یک سیستم، توانایی آن برای اجتناب از نواقص سرویس دهی است که اغلب مکرر بوده و قبول آنها سخت است.

۱-۵. زمان‌بندي

در مدل شبکه پایه به طور کلی از تعدادی میزبان (host) ترکیب شده است، هر ترکیبی از چندین منبع محاسباتی، که ممکن است هماهنگ یا ناهماهنگ باشند، تشکیل شده‌اند. چهار بلوک اساسی ایجاد یک مدل شبکه، کاربر، واسطه منابع، سرویس اطلاعات شبکه (GIS) و آخرین منابع هستند. هنگامی که کاربر نیازمند اجرایی با سرعت بالا است، این عمل به واسطه در شبکه ارائه شده است. شکاف‌های واسطه این عمل دارای وظایف مختلف هستند و در منابع مختلفی بر طبق نیاز کاربرد و منابع در دسترس، توزیع شده‌اند. GIS موقعیت اطلاعات تمام منابعی که به واسطه برای زمان‌بندی کمک می‌کند را حفظ می‌کند.

عمل زمان‌بندی: عمل زمان‌بندی، نگاشتی (مسیردهی‌ای) از اعمال برای منابع فیزیکی مخصوص است، که سعی در به حداقل رساندن هزینه برخی کاربردهای مخصوص استفاده شده توسط کاربر را دارد. و این یک مسئله ناتمام و ابتکاری‌ای است که ممکن است برای رسیدن به راه‌حلی بهینه یا نزدیک به بهینه به کار گرفته شود. محاسبات مؤثر و عمل زمان‌بندی به سرعت یکی از مسائل چالش‌برانگیز در محسبات شبکه‌ای شده و همانطور که دیدید برای موفقیت آن یک امر ضروری است.

زمان‌بندی منابع: فرآیند زمان‌بندی منابع شبکه می‌تواند به عنوان فرایندهای تطابق جستجو برای منابع، شرح اصطلاحات با کاراکترهای لازم، برای تطابق با منابعی که لازم‌اند، باشد. برای در دسترس قرار دادن اطلاعات

بطور سریع برای کاربران و قابلیت اطمینان، یک مکانیسم زمان‌بندی منابع به طور موثر و کارا، بسیار مهم است. به طور کلی منابع شبکه، بطور بالقوه شمار خیلی زیادی با منابع مستقل مختلف است که دارای کنترل مرکزی نیستند. این منابع می‌تواند بخوبی در هر زمانی وارد سیستم شبکه شوند. به همین دلایل زمان‌بندی منابع در مقیاس وسیع شبکه‌ها می‌تواند خیلی چالش‌برانگیز باشد.

۱-۱-۵. زمان‌بندي تلرانس نقص

این استراتژی، تاریخ وقوع نقص در منابع در سرویس اطلاعات شبکه (GIS) را حفظ کرده است. هنگامی که یک واسطه منبعی برای زمان بندی استفاده از تاریخ وقوع نقص منابع (RFOH) اطلاعاتی از GIS و متعاقب آن این اطلاعات دارای استفاده‌های مختلفی در نقطه مقابله‌ای و تکرار در طول عمل زمان‌بندی بر روی منابع است. علاوه بر این، برای افزایش درصد برنامه‌های که به طور مخصوص در یک زمان مشخصی اجرا می‌شوند. با بکارگیری تکنیک‌های نقطه مقابله، این استراتژی می‌تواند زمان‌بندی شبکه‌ای قابل اعتمادتر و کارآمدتری ارائه دهد [۱۳].

این الگوریتم همچنین می‌تواند از اطلاعات RFOH در الگوریتم تکوینی برای برنامه زمان‌بندی با منابع بهینه‌شده استفاده کند [۱۴]. و این می‌تواند احتمال انتخاب این منابع به عنوان اینکه دارای تاریخ بیشتری از رخدادهای ناقص هستند را کاهش دهد. بنابراین ما زمان‌بندی قابل اطمینان و نوعی از تلرانس نقص را داریم. در این مقاله ما GA تازه‌ای برای عمل زمان‌بندی قابل اطمینان در شبکه را نشان دادیم. این الگوریتم از اطلاعات RFOH که در FOHT ذکر شده‌اند را بکار گرفته است. بکارگیری این اطلاعات سبب کاهش شانس انتخاب این منابع به عنوان منابع دارای احتمال نقص بالا، شده است.

فضای چندتایی شبکه (GRIDTS)، در منابعی که وظایف اجرایی بر عهده دارند، به جای بکارگیری از زمان‌بندی متمرکز، نامتمرکز و تلرانس نقص شبکه زیربنایی است [۱۲]. این ارتباط سبب استفاده از فضای چندتایی می‌شود. که این موجب ترکیب تکنیک‌های تلرانس نقص مختلف، نقطه مقابله‌ای، تبادل، تکرار برای ارائه زمان‌بندی تلرانس نقص، می‌شود.

۵-۲. عامل گرایش به تلرانس نقص (عامل متحرك)

در [۶]، عوامل محترك برای تهیه‌ی تلرانس نقص بکار برده شده است. این مکانیسم به نام MAG (تکنولوژی عامل متحرت برای محیط‌های محاسباتی شبکه) نامیده شده است. اکنون، محاسبات تلرانس نقص، همچون عوامل متحرم برای ارائه تلرانس نقص توسعه یافته‌اند. عوامل متحرك از اجتماع عوامل چندگانه است. عامل اطلاعات مربوط به شرایط سخت‌افزار، فرایندهای اجرایی در محاسبات حافظه، منابع دردسترس، شرایط شبکه و زمان متوسط نقص اجزا را نگهداری می‌کند. بر اساس این اطلاعات و وضعیت‌های بحرانی، عامل، سیستم شبکه را قادر به تلرانس نقص می‌کند.

۵-۳. تلرانس نقص در ذخیره‌سازي پیشرفته

ذخیره‌سازی پیشرفته فرایندهایی است که نیازمند منابع مختلفی برای استفاده از زمان گذشته است. که روش‌های کارایی برای ضمانت در دسترس بودن منابع برای کاربران و عملیات در زمان لازم، ارائه می‌دهد. که سبب افزایش احتمال پذیرش عمل و برنامه‌ریزی منابع برای استفاده‌ی مجاز کاربران در سطوح پیشرفته برای رسیدن به دستیابی همزمان کاربردهای آن و اجرای موازی آنهاست.

در [۷]، بازیابی نواقص تنها برای بکارگیری برنامه‌های فعال قبلی نیست، بلکه همچنین برای آنهایی که پذیرفته شده‌اند اما هنوز شروع نشده‌اند، برای برنامه‌های غیرفعال نیز هست. این بدان معنی است که تأثیر برنامه‌های

غیرفعال باید دوباره در منابع پیشرفته دیگر، طرح‌ریزی شود. در طول اجرا هر گونه نقصی که در گره‌ها رخ می‌دهد تأثیر فعالی بر روی برنامه‌هایی می‌گذارد که برای منابع دیگر در نظر گرفته شده‌اند و از تکنیک طرح‌ریزی دوباره در هر یک از آنهایی که زمانشان به پایان رسیده است استفاده می‌کنند، اما این باعث افزایش شمار برنامه‌های پایان یافته می‌شود. طرح‌ریزی دوباره برنامه‌های پذیرفته شده اما هنوز فعال نشده، برای کاهش شمار برنامه‌های پایان یافته و بنابراین رویهمرفته بکارگیری منابع لازم است.

پیوند معیوب می‌تواند بوسیله جریان‌های مسیریابی بر اساس بار واقعی در شبکه قابل بازیابی است. هنگامی که نقص موجود در شبکه اخطار داده می‌شود، جریان‌های مؤثر بسوی مسیرهای جایگزینی استفاده از استراتژی مستقل- زمان توقف، طرح‌ریزی شده است [۹].

یک برنامه اضافه (بیش از زمان مشخص شده) ممکن است به سمت برنامه‌های غیرعادی پایانی متمایل شود (نقص فرآیند). که این بر روی منابع بهره‌برداری از برنامه‌های تأثیر خواهد گذاشت. این سیستم می‌تواند بعدا برنامه‌های غیرفعال را به سمت دیگر گره‌های تبادل هنگامی که آن برنامه فعال است دوباره بررسی کند. بنابراین سبب کاهش نسبت خاتمه غیرنرمال و بهبود عملکرد سیستم می‌شود.

۴-۵. تراز بار تلرانس نقص

تراز بار، تکنیکی برای افزایش منابع، بکارگیری موازی، بهره‌برداری در طول بدیهه‌سازی، و حذف زمان پاسخگویی بوسیله توزیع مناسب کاربردها است. این مدل [۱۹] می‌تواند وظایف نقطه مقابله‌ای در موقعیتی سازگار و انتقال این وظایف بلافاصله به سطوح بالاتر گره‌ها و استفاده از استراتژی تراز بار گروه داخلی و شبکه داخلی، باشد. این استفاده‌های یابنده‌ی نقص که موجب یافتن نواقص منابع موجود و مدیریت نقص می‌شوند و تضمین می‌کنند که وظایف ارائه شده به طور کامل اجرا شده و از منابع در دسترس استفاده شود. در این مدل

تراز بار، اگر گره کارگر N_{ij} خراب شود، سپس در سطح اول: یابنده نقص (FD) در این گروه N_{ij} خراب شده را یافته و به مدیر نقص (FM) گزارش می‌دهد و FM تصمیم می‌گیرد که آیا شروع کند یا نه اینکه یک عامل مهاجرتی وظیفه محلی از N_{ij} برای استراحت این گره‌های کارگر بدهد و در سطح دوم: وظایف N_{ij} به دیگر گروه‌های بار واگذار شود. (حداقل هزینه ارتباطاتی برای کارهای انتقال).

۵-۵. تلرانس نقص و بازیابي جریان‌هاي کار

شبکه جریان کار به عنوان ترتیب هماهنگی از مجموعه وظایف ذره‌ای فرایندهای شرکت کننده در منابع تعریف شده است. اخیراً، نمودار غیرحلقه‌ای مستقیم (DAG) بطور وسیعی در مدل‌های جریان کار محاسبات علمی بکار گرفته شده است. در محیط شبکه‌ای، جریان کار نواقص اجرایی می‌تواند به دلایل مختلفی رخ دهد: تنوع پیکربندی در محیط اجرا، در دسترس نبودن سرویس‌های موردنیاز یا اجزای نرم‌افزاری، بارزیادی شرایط منابع، اجرای خارج از حافظه سیستم، و نقص در اجزا و اجزاء اساسی شبکه. سیستم‌های مدیریت جریان کار شبکه باید قادر به شناسایی و بررسی نواقص و حمایت از عملیات معقول همزمان با نواقص باشند. هوانگ و دیگران [۳۲] تکنیک‌های بررسی نواقص جریان کار را به دو سطح مختلف تقسیم کرده‌اند، به نام سطح وظیفه و سطح جریان کار که ممکن است در نتیجه اجرا و مرتب کردن آدرس نواقص، تغییر کند. هوانگ و کسلمان سه تکنیک متفاوت را پیشهاد دادند. i) تکنیک وظیفه یدکی که دیگر وظایف قطعی را اجرا می‌کند اگر قبلاً نقصی رخ داده باشد. ii) تکنیک افزونگی که وظایف جایگزینی چندگانه‌ای به طور مشابه انجام می‌دهد. iii) انتظار تعریف شده توسط کاربرد برای اجازه بررسی به کاربران جته تعیین یک تهدید مشخص برای نواقص مشخص در جریان کار. در سطح جریان کار، نواقص ممکن است در جریان تحرکات داده یا حلقه‌های نامحدود در جریان کار پویا، اتفاق بیفتند. داده ورودی غلط یا غیرقابل دسترس نیز می‌تواند سبب ایجاد نقص شود.

۶-۵. تلرانس نقص در تخصیص منابع

یکی از انتظارات از محاسبات شبکه این است که قادر به اجرای اجرای درخواست‌ها در طول مکان‌های مختلف باشد. برخی از این درخواست‌ها نیازمند هماهنگی دستیابی به منابع مدیریت شده توسط موجودیت‌های مستقل است. این هماهنگی دستیابی به عنوان تخصیص منابع شناخته شده است. تخصیص منابع فرایندی است برای تخصیص منابع از تهیه کنندگان چندگانه به صورت حالتی همانگ شده است. شبکه‌ی محاسباتی به طور بخصوص ترکیبی از چندین بخش از سازماندهی‌های شرکت کننده از لحاظ جغرافیایی است که برنامه‌های موازی باید برای منتشر کردن در بیشتر از یک محل به طور همزمان در چندی محل بدون ملاحظه محدودیت منابع از یک محل مستقل، زمان‌بندی شوند. این الگوریتم زمان‌بندی شبکه باید قادر به هماهنگ کردن این منابع از بخش‌های مختلف باشد [۳۰].

۷-۵. مکانیسم تلرانس نقص براي مدیریت منابع آگاه- SLA

توافقنامه سطح سرویس (SLA) یک ابزار قدرتمندی برای توضیح تمام انتظارات و التزامات در روابط کاری مابین سرویس مشتری و سرویس تهیه کننده، است. این تنها پوششی برای سؤالات و درخصوص منابع لازم نیست، بلکه برای موضوعاتی از قبیل QOS (کیفیت سرویس خدمات)، تلرانس نقص، یا اندازه‌گیری میزان SLA نیز بکار می‌رود. بعلاوه، شامل ارزش زیادی برای محاسبات منابع، مخصوصاً جریمه ورودی برای شکستن توافق‌نامه است. این سیستم به شبکه اجزاه می‌دهد که بر روی SLAها مذاکره کند، پایبندی به SLAهای پذیرفته شده توسط نقطه مقابل عملیات- انتقال و مهاجرت، را تضمین کند. همانند ذخیره‌سازی پیشرفته، انتشار تقاضاها، و مذاکره درخصوص پروتکل‌هایی که الزاماً وابسته به آگاهی- SLA هستند RMS (سیستم مدیریت منابع). RMS دسترسی شفاف و یکسانی برای بخش اعظمی از منابع ناهماهنگ ارائه می دهد [۱۹].

در منابع شبکه نواقص- آگاه سیستم مدیریت، مدیر منابع مجازی (VRM) است که از QOS بوسیله SLAها حمایت می‌کند [۷]. بطور عمده بر روی برنامه‌های تمرکز دارد که برای منابع نقصان یافته زمان‌بندی شده‌اند و هنوز زمان اجرای آنها شروع نشده است، که به همین دلیل برنامه‌های غیرفعال نامیده می‌شوند. الگوریتم‌های نقطه مقابل استفاده شده در این مقاله، بر اساس این مفهوم بوده‌اند و بنابراین دارای اشتراک توافقی هستند.

۸-۵. تلرانس نقص در شبکه نوري

در شبکه نوری [۲۱]، این کاربردها به عنوان DAG مدل‌سازی شده‌اند. هر وظیفه می‌تواند به عنوان یک نقطه مقابله‌ای نشان داده شود. اگر منابع شبکه دچار نقص شوند، این کاربرد می‌تواند از وظایف متناوب به جای اجرای مجدد کاربرد، ادامه یابد. منابع شبکه شامل محاسبات، ذخیره و دستگاه‌های تجسم‌زای بوسیله شبکه نوری به و با سویچ‌های نوری و پیوندهای نوری به هم مرتبط شده‌اند. هنگامی که نقصی برای منابع شبکه یا در یک پیوند نوری رخ می‌دهد، کاربردهای دیگر وظایف در منبع یا داده ارتباطات در پیوند نوری، گسیخته و متوقف می‌شوند. این نقص مستقل از دیگران است. این نقص در سیستم شبکه نوری مدیریت می‌شوند، و تمام نواقص می‌توانند خیلی زود پیدا شوند. در این مقاله رویه‌های زمان‌بندی مجدد برای دستیابی به تلرانس نقص در شبکه نوری ارائه شده است.

در طول اجرا، اگر نقصی رخ دهد، مدیر نقص قادر به تحصحیح اطلاعات نقصان یافته و مدیر اجرایی قادر به تصحیح اطلاعات اجرای کاربرد است. بر اساس این اطلاعات درخصوص نقص و کاربرد، می‌توان وظیفه ناتمام و داده ارتباطی را دوباره زمان‌بندی کرد. اگر که وظیفه اجرایی توسط نقص منبع گسیخته شده است، باید دوباره اجرا شود، اما برای ارتباط داده‌ای قطع شده می‌توان از نقطه قطع شده بجای نقطه آغاز داده، شروع کرد. در یک وظیفه گسیخته شده که زمان‌بندی برای دیگر منابع صورت گرفته است، تمام داده‌ها لازم است که به منبع انتقال

داده شوند. این الگوریتم زمان‌بندی مجدد، قسمت‌های ناتمام را کاربردها را بر پایه اطلاعات نقصان یافته زمان‌بندی می‌کند، بنابراین قادر به یافتن زمان‌بندی بهتر برای کاربردها در زمان اجرای کمتری است.

نتیجه‌گیری

این مطالعه پیشرفت‌های صورت گرفته در سیستم‌های شبکه‌ای با قابلیت اطمینان بالا را بررسی کرده است. می‌توان گفت، برای داده. تلاش برای ایجاد سیستم‌های شبکه‌ای با قابلیت اطمینان بالا مبتنی بر روش‌های پیشرفته برای تلرانس نقص ایجاد شده است. تلاش‌های درخصوص بهبود تلرانس نقص بر مناطق کاربردی مشخص در محاسبات شبکه‌ای متمرکز شده است، شامل محاسبات منابع سخت‌افزاری و نرم‌افزاری، کاربردهای کاربر و جریان کار، زمان‌بندی، ذخیره‌سازی پیشرفته، و شبکه های در هم تنیده. بطور کلی، بر روی فرآیند های متفاوت در این مناطق، و در هر منطقه صورت گرفته است، مسائل مختلف حل شده‌اند.

بنابراین، روش‌های عیب‌یابی جدید، سرویس شفاف تلرانس نقص معماری، بر اساس تکنیک‌های تلرانس نقص، مدل تلرانس نقص اقتصادی، سیستم پیش‌بینی نقص نوری، مدل تلرانس نقص چندگانه و خودانطباقی چهارچوب تلرانس نقص، برای ایجاد محیط شبکه‌ای پیشنهاد شده است که بیشتر قابل اطمینان و درست است.

قابلیت اطمینان نرم افزاری در سیستم‌های پیچیده، موضوع مهمی برای شرکت‌های بزرگ با اعتبار مسلم است. برای جلوگیری از، خسارات پولی و از دست دادن مشتریان، داشتن سیستم تلرانس نقصی که قادر به کپسوله‌سازی بسیاری از توانایی‌ها، که نهایت در خدمت جلوگیری از نواقص سیستم موجود هستند، مهم است.

مقاله حاضر به دو بخش تقسیم شد. در قسمت اول ما توضیحاتی در خصوص مکانیسم‌های کلی استفاده شده در نواقص، که به استفاده از رویکرهای خودشفادهنده و خودسازمان‌دهنده در دستیابی به اهداف ما می‌پردازند، ارائه دادیم. بخش دوم این مقاله بر روی طراحی تلرانس نقص در سیستم چندعاملی متمرکز شده است که خصوصیات آن در بخش اول این مقاله توضیح داده شده و مورد تجزیه و تحلیل قرار گرفته است.

بخش بعدی این پروژه درصدد اجرای رویه‌ها و اهداف معماری همراه با جزئیات توضیح داده شده در محیط Jad است. دیگر توضیحات ممکن درصدد نشان دادند نواقص در مکانیسم‌های خودی با استفاده از مکانیسم‌های مانند بکارگیری بازبینی محاسبات یا تکرار اجزاء کلیدی خودشفادهنده است.

منابع

[1] Ian Foster and Carl Kesselman, S.T., (2001) “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, Intl J. Supercomputer Applications, 15: 200-222.

[2] I. Foster and C. kesselman, (2004) “The Grid: Bluprint for a New Computing Infrastructure, 2nd Edition, and pp: 748.

[3] R. Medeiros, W. Cirne, f. Braso;erop amd J. Sauve, (2003) “ Faults in Grids: Why are they so bad and What can be done about it?, In Proceedings of the Fourth International Workshop on grid Computing.

[4] Weissman, J.B, (1999) “Fault Tolerant Computing on the Grid”, pp: 351-352, HPDC.

[5] Mohammad Tanvir Huda, Heinz and W. Schmidt, Ian D. Peake , (2005) ”An Agent Oriented proactive Fault-tolerant framework for Grid Computing”, In Proceedings of the First IEEE International Conference on e-Science and Grid Computing, pp.304-311.

[6] Rafael Fernandes Lopes and Francisco Jos´e da Silva e Silva, (2006) “Fault tolerance in a Mobile Agent Based Computational grid”, In Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops.

[7] Lars-Olof Burchard, Cesar, A. F. De Rose, Hans-Ulrich Heiss, Barry Linnert, and Jorg Schneider, (2005) “VRM: A Failure-Aware Grid Resource Management System”, In Proceedings of the 17th IEEE International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD’05) 1550-6533.

[8] Buyya R, (2002) “ Economic-based distributed resource management and scheduling for grid computing Ph.D. Paper, Monash University, Melbourne, Australia.

[9] Lars-Olof Burchard, Barry Linnert and Joerg Schneider, (2005) “Rerouting Strategies for Networks with Advance Reservations”, Proceedings of the First IEEE International Conference on e-Science and Grid Computing 0-7695-2448-6.

[10] Nazir, B. and Khan, T., (2006) “Fault Tolerant job Scheduling in Computational Grid. Emerging Technologies, In IEEE International Conference on Emerging Technologies, Volume, Issue, 13-14.

[11] Libing Wu1, Chanle Wu1 , Jianqun Cui , Huyin Zhang1 and Gang Ye, (2006) ” An Overtimetolerance Strategy for Advance reservation”,in Proceedings of the Seventh international Conference on Parallel and distributed Computing, Applications and Technologies.

[12] Fabio Favarim, joni da silva Fraga, lau Cheuk Lung and Miguel Correia,(2007) “GRID TS: A New Approach for Fault-Tolerant Scheduling in Grid Computing”, In International Symposium on Network Computing and applications, pages: 187-194.

[13] Leyli Mohammad Khanli, (2010) “Reliable Job Scheduler using RFOH in Grid Computing”, Journal of Emerging Trends in Computing and Information Sciences, Vol. 1, No. 1.

[14] Leili Mohammad Khanli , Maryam Etminan and Far Amir Masoud Rahmani , (2010) ”RFOH: A New Fault Tolerant Job Scheduler in Grid Computing”, In Second International Conference on Computer Engineering and Applications .

[15] P. Stelling, I. Foster, C. Kesselman, C. Lee and G. Von Laszewski, (1998) “A fault detection service for wide area distributed computations”, In Proceedings of 7thIEEE symposium on High Performance Distributed Computing.

[16] Babar Nazir, Kalim Qureshi and Paul Manuel, (2009) “Adaptive checkpointing strategy to tolerate faults in economy based grid”, J Supercomput , 50: 1-18.

[17] Lars-Olof Burchard’, Matthias Hovestadt’, Odej Kao’, Axel Keller’ and Barry Linnert’,(2004)“The virtual Resource Manager: An Architecture for SLA-aware Resource Management”, in IEEE International Symposium on Cluster Computing and the Grid.

[18] Matthias hovestadt, (2005) ”fault tolerance Mechnisms for SLA-aware Resource management”, in International conference on Parrallel and Distributed Systems.

[19] B. Yagoubi and M. Medebber, (2007) “A Load Balancing Model for Grid Environment”, IEEE.

[20] Elvin Sindrilaru, Alexandru Costan , Valentin Cristea , (2010) “Fault Tolerance and Recovery in Grid Workflow Management Systems”, In International Conference on Complex, Intelligent and Software Intensive Systems.

[21] Y. Wang, Y. H. jin, W. Guo, W. Q. Sun, W. S. hu and M. Y. Wu, (2007) “ Joint Scheduling for Optical Grid Applications”, Journal of Optical Networking, Vol. 6, pp. 304-318.

[22] J. H. Abawajy, (2004) “ Fault-Tolerant Scheduling Policy for Grid Computing Systems”, In Proceedings of the 18th International Parallel and Distributed Processing Symposium.

[23] Satish Tadepalli, Calvin Ribbens and Srinidhi Varadarajan, “GEMS: A Job Management System for Fault Tolerant Grid Computing”, High-Performance Computing Symposium, ISBN: 1-56555-278-4.
[24] Eric roman, (2002) “A Survey of checkpoint/restart Implementations”, Lawrence Berkley National Laboratory, CA.
[25] Paul Townend, jie Xu, (2003) ”Fault tolerance within a grid Environment”, As part of the e-Demand project at the University of Durham, DHI 3LE, United Kingdom.
[26] A. Nguyen-Tuong, (2000) “Integrating fault-tolerance techniques in Grid applications”, Ph.D, Dissertation, university of Virginia.
[27] J. Daly, (2003) “A model for predicting the optimum checkpoint interval for restart dumps”, in Proceedings of the ICCS2003, Lecture Notes in Computer Science 2660, Vol. 4, pp:3-12.
[28] H. Lee, K. Chung, S. Chin, J. Lee, D. Lee, S. Park and H.Yu, (2005) “ A resource management and fault tolerance services in grid computing”, J Parallel Distributed Computing, vol. 65(11), pp.1305-1317.
[29] Antonios Litke, Dimitrios Halkos, Konstantinos Tserpes, Dimosthenis Kyriazis and Theodora Varvarigou, (2009) “ Fault tolerant and prioritized scheduling in OGSA-based mobile Grids”, Concurrency and computation practice and experience, 21:533-556.
[30] Foster I , Kesselman C, Lee C, Lindell B, Nahrstedt K and Roy A,(1999) “A distributed resource management architecture that supports advance reservation and co-a allocation”, Proceedings of the International Workshop on QoS, London, U.K.,27-36.
[31] Eduardo Huedo, Ruben S. Montero, Ignacio M. Llorente, (2006) ”Evaluating the reliability of computational grids from the end user’s point of view”, Journal of Systems Architecture, 727-736.
[32] S. Hwang and C. Kesselman, (2003) “Grid Workflow: A Flexible Filure handling Framework for the Grid”, In Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing, Seattle, Washington, USA.
[33] Israel Koren and C.Mani Krishna, (2007) “Fault Tolerant Systems”, Elsevier Inc., ISBN: 978-81-312-1530-2.
[34] Upadhyaya Debanjan Ghosh H. Raghav Rao Shambhu and Raj Sharman. Self-healing systems - survey and synthesis. Decision Support Systems, 42(4):2164{2185, January 2007.
[35] G. Caire. D. Greenwood F. Bellifemine. Developing Multi-Agent Systems with Jade. Wiley Series in Agent Technology, 2007.

[36] M.-P. Gleizes G. D. M. Serugendo and A. Karageorgos. Self-organization in multi-agent systems, volume 20(2):165-189. The Knowledge Engineering Review, 2005.
[37] Schahram Dustdar Harald Psaier. A survey on self-healing systems: approaches and systems. Computing, pages 91(1):43{73. August 2010.
[38] Christian Koppen Hermann De Meer. Characterization of self-organization. in peer- to-peer systems and applications. 2005.
[39] McCann JA Huebscher MC. A survey of autonomic computing - degrees, models, and applications. 40(3):1-28, August 2008.
[40] Yves Crouzet Yves Deswarte Jean-Charles Fabre Jean-Claude Laprie Jean Arlat and David Powell. Tolerance aux fautes., pages 241{270. In Encyclopedie de l'Informatique et des Syst`emes d'Information, 2006.
[41] M. Kowalczyk W. Jelasity and M. van Steen. Newscast computing. Technical report, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, 2003.
[42] Patrick Taillibert Katia Potiron and Amal El Fallah Seghrouchni. A step towards fault tolerance for multi-agent systems., volume 5118/2008 of In Languages, Methodologies and Development Tools for Multi-Agent Systems, pages 156{172. Springer Berlin / Heidelberg, France, 2008.
[43] Massimo Paolucci. Martin Van Velsenv-Joseph Giampapa Katia Sycara. The retsina mas infrastructure. in autonomous agents and multi-agent systems. volume 7, pages 29{48. Springer Netherlands, Pittsburgh, 2004.
[44] Rachid Guerraoui Anne-Marie Kermarrec, Mark Jelasity and Maarten van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based imple-mentations. volume 3231 of In Lecture Notes in Computer Science. Springer-Verlag,2004.
[45] Ozalp Babaoglu Mark Jelasity. T-man: Gossip-based overlay topology management. volume 3910 of In Engineering Self-Organising Systems, pages 1{15. Springer Berlin / Heidelberg, 2006.
[46] Cynthia Nikolai and Gregory Madey. Tools of the trade: A survey of various agent based modeling platforms. Arti_cial Societies and Social Simulations. 2009.
[47] Tahvildari L Salehie M. Self-adaptive software: landscape and research challenges, pages 4(2):1{42. ACM Transactions on Autonomous and Adaptive Systems (TAAS), May 2009.
[48] Y. Vigfusson K. Birman A.-M. Kermarrec V. Gramoli and R. van Renesse. Sliver, a fast distributed slicing algorithm. Master's thesis, Cornell University, 2007.

[49] Durfee E.H. Lesser V.R. and D.D. Corkill. Trends in cooperative distributed problem solving. volume KDE-1(1):63-83. IEEE Transactions on Knowledge and Data Engi-neering, March 1989.
[50] Michael Winiko_. JackTM intelligent agents: An industrial strength platform, vol- ume 15. Springer US, 2005.