دانشکده مهندسی کامپیوتر
دانشگاه صنعتی شریف

گزارش سمینار
Report of Seminar of
شبکه های میان ارتباطی
Interconnection Networks

مدل تحلیلی برای چندپردازنده های با حافظه مشترک

Analytical Model for Shared Memory Multiprocessors

استاد: دکتر سربازی آزاد
Prof.: Dr. Sarbazi Azad

وحید مواجی
Vahid Mavaji

تیر ۸۴
July 2005

فهرست مطالب

مقـــدمـــــه

محاسبات با کارایی بالا در بسیاری از کاربردها مثل شبیه سازی پدیده های فیزیکی، شبیه سازی مدارات مجتمع و شبکه های عصبی، مدل سازی هوا، آئرودینامیک و پردازش تصویر، مورد نیاز است. این محاسبات بطور فزاینده ای متکی به گره های کامپیوتری می باشد که گروه هایی از آنها با یک شبکه میان ارتباطی به هم متصل می شوند تا یک سیستم چندپردازنده با حافظه توزیع شده را تشکیل دهند. چنین سیستم هایی مقیاس پذیرند و توان محاسباتی بالایی دارند. پردازه ها در گره های مختلف از طریق ارسال پیغام با یکدیگر ارتباط برقرار می کنند. برنامه نویس ها باید از دستورات send/receive استفاده کنند و توزیع داده ها را بطور صریح و دستی مدیریت کنند، کاری که با افزایش اندازه برنامه، بسیار پیچیده خواهد شد. فرموله کردن و حل بسیاری از برنامه های موازی، با استفاده از روش حافظه – مشترک، راحت تر از ارسال پیغام می باشد. سیستم های حافظه – مشترک سنتی، یک مدل برنامه نویسی عام و راحت، بدست می دهند ولی چون پردازنده ها به شدت به هم متصل هستند، با افزایش اندازه، این سیستم ها دچار ازدحام زیاد و تأخیر بیشتر می شوند.

یک سیستم حافظه – مشترک توزیع شده(DSM) [۱] را می توان بصورت مجموعه ای از گره ها یا کلاسترها در نظر گرفت که از طریق یک شبکه میان ارتباطی با هم ارتباط برقرار می کنند. این سیستم، مکانیزم ارسال پیغام را پنهان می کند و یک مدل حافظه – مشترک فراهم می آورد که راحتی برنامه نویسی و تصادم [۲] کاهش یافته را با هم ترکیب می سازد. DSM به عامل های [۳] مدیریت متکی است که از ارسال پیغام برای نگاشت فضای آدرس منطقی مشترک به حافظه های محلی استفاده می کنند و آنرا همواره منسجم [۴] نگه می دارند. در هر دسترسی به فضای مشترک، سخت افزار باید تعیین کند که آیا داده درخواست شده در حافظه محلی می باشد و اگر نبود، داده باید از حافظه دور [۵] آورده شود. برای موقعی که داده در فضای مشترک نوشته می شود نیزعملیاتی لازم است تا انسجام داده های مشترک حفظ شود.

یک هدف مهم از تحقیقات روی سیستم های DSM، توسعه راهکارهایی است که با حفظ سازگاری [۶] داده ها، زمان دسترسی به داده مشترک را به حداقل برسانند. کارایی [۷] آنها به این بستگی دارد که مدل سازگاری داده ها، چقدر محدود کننده است. مدل هایی که محدودیت

های قوی دارند، منجر به تأخیر دسترسی زیاد و پهنای باند بیشتر می شوند. مدل های پیشرفته تر با قیود ضعیف تر، ارائه و پیاده سازی شده اند. این مدل ها اجازه مرتب سازی دوباره[۸]، استفاده از خط لوله[۹]، و همپوشانی حافظه[۱۰] را می دهند و نهایتا کارایی بهتری دارند ولی نیازمند این هستند که دسترسی به داده های مشترک به طور صریح، همزمان سازی[۱۱] شود که منجر به درگیری بیشتر برنامه نویس و عدم راحتی در برنامه نویسی می گردد. چنین مدل هایی مفید می باشند ولی نیاز به تلاش اضافه از طرف برنامه نویس، مسائلی را پیش می آورد که به خاطر آنها از روش ارسال پیغام اجتناب می کردیم. موفقیت DSM به قابلیت رها ساختن برنامه نویس از هر عملی که فقط برای همخوانی با مدل حافظه ضروری می باشد، بستگی دارد.

بررسی و مدل کردن سیستم های حافظه – مشترک بر اساس پارامترها و معیارهای گوناگونی انجام می شود. در این گزارش ما چند مورد خاص را بررسی می کنیم. در هر مورد، مفروضات مدل، گفته شده و بر اساس این مفروضات، مدل تحلیلی ارائه شده است. با اینکه حالت های گوناگونی برای مدل کردن می توان در نظر گرفت ولی چهارچوب کلی برای همه یکسان است یعنی با استفاده از مطالب این گزارش و روش کلی تحلیلی که ارائه شده است، می توان معماری های حافظه – مشترک دیگر را هم مدل نمود.

بررسی مقالات

تأثیر ویژگی های شبکه میان ارتباطی و پروتکل های سازگاری، موضوع تحقیقات گسترده ای بوده است. یک چندپردازنده DSM مبتنی بر مش دو بعدی با استفاده از مدل شبکه های صف و شبیه سازی در [۱] بررسی شده است. برای مقادیر بالای احتمال درخواست از یک حافظه دور، آنها یافته اند که شبکه میان ارتباطی اشباع می شود و بازدهی[۱۲] پردازنده، زیر ۳۵% می ماند.

هم روش تئوری و هم شبیه سازی در [۲] بکار رفته اند تا یک چندپردازنده DSM کلاسترشده بررسی شود. از شبکه crossbar برای اتصال پردازنده ها و حافظه ها درهر کلاستر و پردازنده ها و حافظه سراسری[۱۳] استفاده شده است. هر چقدر احتمال دسترسی به حافظه در هر سایکل افزایش می یابد، بالاترین کارایی سیستم، تقریبا ۲۵% ماکزیمم ممکن می

شود وقتی که همه دسترسی ها به حافظه سراسری هدایت می شوند و ۶۰% ماکزیمم ممکن می گردد وقتی که همه دسترسی ها به حافظه محلی یا کلاستر یا سراسری با احتمال یکسان هدایت شوند.

کارایی پردازنده در چهار کاربرد فضایی – علمی روی HP/Convex Exemplar بررسی شده است [۳]. این کامپیوتر حداکثر ۱۶ کلاستر دارد(هر کدام چهار پردازنده) که با چهار حلقه SCI به هم متصل شده اند. تسریع کاهش یافته در برخی کاربردها، بخاطر الگوهای نامنظم دسترسی به داده ها، ارتباطات سراسری بین پردازنده ها و توازن بار[۱۴] می باشد.

بررسی چهار معماری با پشتیبانی سخت افزاری حافظه – مشترک در [۴] آمده است. حتی تحت خوشبینانه ترین مفروضات، علی الخصوص در مشخص کردن عدم برخورد در کش[۱۵] تأخیر عمده ای یافت شده است که منجر به ترافیک در شبکه میان ارتباطی می گردد.

یک DSM روی یک nCUBE با ۱۶ گره در [۵] بررسی شده است. آزمایشات با چهار برنامه موازی نشان می دهد که در جمع ماتریسی روی داده های توزیع شده، کارایی کاهش یافته و زمان ارسال داده زیادی در مقایسه با زمان محاسبات درون یک گره مورد نیاز است. مشاهده می شود که چنین برنامه هایی برای DSM نامناسب می باشند مگر اینکه تکنیک هایی ابداع گردد تا از میزان ارتباطات بکاهند.

یک چندپردازنده DSM مبتنی بر شبکه باس چندمرحله ای[۱۶] در [۶] مطالعه شده است. یک پردازنده می تواند یا مشغول باشد یا منتظر پاسخ به یک درخواست حافظه باشد. برای همه مقادیر احتمال درخواست کمتر از ۰,۱، بازدهی پردازنده زیر ۶۵% می ماند وقتی تقریبا همه بسته ها به حافظه محلی هدایت می شوند و زیر ۴۰% می ماند وقتی تقریبا همه بسته ها به حافظه سراسری هدایت می گردند. بازدهی به طور قابل ملاحظه ای با افزایش احتمال درخواست، افت می کند – بدلیل مقادیر بالاتر ترافیک و تأخیر در صف.

یک چندپردازنده مبتنی بر مش ۴*۴ با مسیر یابی wormhole در [۷] مطالعه شده است. کارایی دو روش پیش – واکشی[۱۷] مبتنی بر سخت افزار، با شبیه سازی، ارزیابی شده

است. با استفاده از پنج برنامه محبوب و رایج، عدم برخورد کش در محدوده ۱% تا ۱۰% مشاهده شده است.

حتی بعد از تلاش های گسترده برای تقویت نرم افزار، بسیاری از محققان، کارایی ضعیف یا متوسط را در بسیاری از کاربردهای در اندازه های بزرگ مشاهده می کنند که اغلب بدلیل عدم توازن بار، همزمان سازی، الگوهای نامنظم و دینامیک ارتباطات و ارتباطات multicast, broadcast برای مدیریت DSM می باشد و بار زیادی را روی شبکه میان ارتباطی قرار می دهد. به عقیده آنها، ارتباطات میان پردازنده ای، آن چیزی است که برنامه نویسی موازی را به چالش می طلبد.

شبکه های میان ارتباطی با کارایی بالا(و با پیچیدگی بالا) ارائه شده اند[۸]. شبکه های عملی تری شامل شبکه های مبتنی بر مش با باس های اضافی برای broadcast نیز ارائه شده اند. ابر – مش[۱۸] با سوئیچ توزیع شده crossbar (یک پیاده سازی از شبکه های ابر – گرافی چند بعدی) در [۹] بررسی شده است که احتمالات بلوکه شدن و تأخیر متوسط پیغام ها محاسبه گردیده است. بطور مشابه، یک پیاده سازی نوری از ابر – مش ها با استفاده از crossbar های الکتریکی و نوری در [۱۰] انجام گردیده است. علیرغم این که چند طول موج بکار رفته است، چند فرستنده ممکن است از یک طول موج استفاده کنند.

یک مدل برای تحلیل میزان توازی بین ماژول های حافظه

می توانیم از هر کدام از معیارهای ارزیابی کارایی یک سیستم چندپردازنده استفاده کنیم. برای اینکه مدل و پیاده سازی آنرا قابل کنترل نگه داریم، روی درجه موازی سازی بین ماژول های حافظه تمرکز می کنیم. یعنی می خواهیم تعداد متوسط ماژول های حافظه که بطور همزمان مورد دسترسی قرار گرفته اند را تعیین کنیم. مدل اینگونه است:

۱. سیستم، p پردازنده و m ماژول حافظه دارد. هر پردازنده می تواند درخواست دسترسی به هر ماژول حافظه بفرستد.
۲. دسترسی به ماژول های حافظه، همگام می باشد؛ دو ماژول که به درخواست های حافظه سرویس می دهند، در یک زمان کار خود را شروع کرده و در یک

زمان به اتمام می رسانند. دسترسی به حافظه همواره یک واحد زمانی طول می کشد(یک سایکل حافظه).

۳. پردازنده ها، بی نهایت سریع می باشند. وقتی درخواست یک پردازنده، توسط یک ماژول حافظه انجام شد، آن پردازنده فورا یک درخواست جدید تولید می کند.
۴. پردازنده ها، درخواست های خود را با احتمال مساوی به هر ماژول می فرستند. ماژول انتخاب شده برای یک درخواست جدید، مستقل از ماژول انتخاب شده برای دیگر درخواست ها است.
۵. یک پردازنده فقط یک درخواست معلق[۱۹] در هر زمان دارد.
۶. یک ماژول حافظه فقط به یک درخواست در هر لحظه سرویس می دهد. اگر بیشتر از یک درخواست در صف قرار داشته باشد، در ابتدای سایکل حافظه، ماژول بطور تصادفی یک درخواست را برای سرویس انتخاب می کند. تمام درخواست هایی که درطی سایکل، سرویس داده نشده اند، تا آغاز سایکل بعد در صف قرار می گیرند.

فرض کنید B تعداد متوسط ماژول های حافظه است که در طی یک سایکل حافظه مشغول سرویس دادن به یک درخواست می باشند. بدون توجه به تعداد پردازنده ها و ماژول های حافظه، می توانیم مدل را بصورت یک زنجیره مارکف زمان – گسسته[۲۰] پیاده سازی کنیم که اطلاعات درون هر حالت، شامل تعداد ماژول های حافظه ای است که در ابتدای یک سایکل حافظه، درخواست دارند. بنابراین یک حالت، تعداد ماژول هایی را که در طی سایکل بعدی مشغول خواهند بود نیز نشان می دهد. می توانیم B را با حل حالت پایدار زنجیره مارکف و محاسبه $B = \Sigma (Pr[state\ i] . Number\ of\ busy\ modules\ in\ state\ i)$ بیابیم.

می دانیم که این زنجیره مارکف یک جواب حالت پایدار دارد چون محدود است و حداقل یک حالت خود حلقه[۲۱] دارد. زنجیره، محدود است چون تعداد حالتهایی که درخواست های پردازنده ها می تواند بین ماژول ها توزیع گردد، محدود است. هر حالتی که حداقل، یک درخواست در هر ماژول حافظه داشته باشد، خود حلقه است چون هر پردازنده ای که درخواستش در طی یک سایکل اجابت شده است، با شروع از این حالت، با یک احتمال غیر صفر، درخواست بعدی خود را به همان ماژول می فرستد. ابتدا برای یک p و m خاص،

مجموعه همه حالت های زنجیره مارکف را تعیین می کنیم. سپس احتمالات تمام انتقالات تک مرحله ای[۲۲] را محاسبه می کنیم. نهایتا زنجیره مارکف را برای احتمالات حالت پایدار حل می نماییم.

درحالت کلی، پیاده سازی مدل با حالتهایی که فقط تعداد ماژول های مشغول در سایکل بعدی را بدست می دهند، ممکن نیست. مثلا مدل یک سیستم با ۴ پردازنده و ۲ ماژول حافظه را درنظر بگیرید. حالتی که فقط می گوید ۲ ماژول در طی سایکل بعدی مشغول خواهند بود، اطلاعات کافی درباره احتمال بودن در حالتهای گوناگون در آغاز سایکل بعدی را به ما نمی دهد. این امر به این دلیل است که توزیع درخواست بین ماژول ها دو حالت دارد:

۱. سه درخواست در یک ماژول، یک درخواست در ماژول دیگر.
۲. دو درخواست در یک ماژول، دو درخواست در ماژول دیگر.

اگر یک سایکل را با چهار درخواست توزیع شده بصورت (۱) آغاز کنیم، در انتهای سایکل، دو درخواست باقیمانده در یک ماژول و هیچ درخواست در ماژول دیگر داریم و دو پردازنده آماده تولید درخواست های جدید هستند. سایکل بعدی می تواند با توزیع چهار درخواست بصورت (۱) یا (۲) یا با چهار درخواست در یک ماژول آغاز گردد. اگر سایکلی را بصورت (۲) آغاز کنیم، هر دو ماژول درخواستی را در انتهای سایکل خواهند داشت. سایکل بعدی نمی تواند با هر چهار درخواست در یک ماژول آغاز گردد.

ما باید تعریفی از حالت داشته باشیم که علاوه بر گفتن این که چند ماژول درطی سایکل مشغول هستند، بگوید چگونه درخواست هایی که باید درطی یک سایکل منتظر بمانند، در انتهای سایکل بین ماژول ها توزیع شده اند. (۱) و (۲) در بالا دقیقا چگونگی این امر را نشان می دهند. هر حالت مشخص می کند که چگونه p درخواست بین ماژول ها توزیع شده است بدون توجه به این کدام ماژول، کدام است. در (۱) اهمیت ندارد بدانیم که کدام ماژول ۳ درخواست دارد و کدام ۲ درخواست، چون پردازنده ها و ماژول ها یکسان هستند.

با این تعریف کلی از حالت، محاسبه احتمالات انتقال حالت می تواند به دو قسمت تقسیم شود. اگر $r \leq p$ درخواست در طی یک سایکل حافظه سرویس داده شوند، r پردازنده ای که درخواست های جدید برای سایکل حافظه بعدی دارند، هر مجموعه خاصی از ماژول

ها را با احتمال $(1/m)^r$ انتخاب می کنند. همه آنها ممکن است درخواست جدید خود را به یک ماژول بفرستند یا این که هر کدام به ماژولی متفاوت از بقیه بفرستند یا بعضی به یک ماژول بفرستند و بقیه ماژول های متفاوتی انتخاب کنند. نتیجه نهایی، توزیعی از p درخواست بین m ماژول در ابتدای سایکل بعدی خواهد بود که هر توزیع ممکن با یک حالت نشان داده می شود.

دومین قسمت محاسبه احتمالات انتقال حالت، شمارش تعداد راه هایی است که پردازنده ها درخواست های جدیدی تولید می کنند که به توزیع یکسانی از درخواست ها می انجامد. مثال زیر با ۴ پردازنده و ۲ماژول حافظه را در نظر بگیرید. فرض کنید مدل، یک سایکل را با دو درخواست در هر ماژول شروع می کند. در انتهای سایکل، هر ماژول یک درخواست باقیمانده دارد. دو پردازنده، قبل از آغاز سایکل بعد، درخواست های جدیدی را تولید می کنند. اگر آنها یک ماژول را انتخاب کنند، حالت بعدی، سه درخواست در یک ماژول و یک در خواست در ماژول دیگر خواهد داشت. دو راه برای این کار وجود دارد، چون دو ماژول وجود دارد که آنها می توانند انتخاب کنند و هر کدام از این ماژول ها، یک درخواست باقیمانده از سایکل قبلی دارد.

برای کنار هم قرار دادن این دو قسمت، احتمال انتقال حالت تک مرحله ای را برای رفتن از حالت A به حالت B با ضرب کردن احتمال انتخاب هر مجموعه از ماژول ها در سایکل شروع شده از A در تعداد راه هایی که می توانیم مجموعه ای از ماژول ها را انتخاب کنیم که ما را به B ببرد، محاسبه می کنیم. برای این مثال، احتمال انتقال از حالتی با دو درخواست در هر ماژول به حالتی با سه درخواست در یک ماژول و یک درخواست در ماژول دیگر برابر $1/2 = 2.(1/2)^2$ می باشد.

بطور کلی فرایند محاسبه احتمالات انتقال حالت می تواند بسیار مشکل باشد. مشکل کار همواره در قسمت دوم است – شمارش تعداد راه های انتخابی که مدل را به حالت بعدی یکسانی می برد. ما کل پروسه با چند مثال تشریح می کنیم. فرض کنید $C(n, r)$ تعداد ترکیب های r شیء از n شیء و $P(n, r)$ تعداد جایگشت های r شیء از n شیء باشد.

مثال ۱: دو پردازنده و $m \geq 2$ ماژول.

	حالت
۲ درخواست در یک ماژول	۱
۱ درخواست در هر ماژول	۲

ابتدا، احتمالات انتقال تک مرحله ای را می یابیم. هر احتمال را بصورت حاصلضرب احتمال رفتن درخواست جدید به یک مجموعه خاص از ماژول ها ($1/m$ اگر سایکل در حالت ۱ شروع شود و $1/m^2$ اگر سایکل در حالت ۲ شروع شود) در تعداد چنین مجموعه هایی که به حالت بعدی یکسان منجر می شوند، می نویسیم.

$p_{11} = (1/m).1 = 1/m$

$p_{12} = (1/m).(m - 1) = (m - 1)/m$

$p_{21} = (1/m^2).C\ (m, 1) = 1/m$

$p_{22} = (1/m^2).C(m, 2) = (m - 1)/m$

چون می دانیم که ماتریس P تکین[۲۳] است، فقط به یکی از معادلات Chapman – Kolmogorov نیاز داریم و لذا فقط به دو احتمال انتقال تک مرحله ای نیاز داریم – یا p_{11}, p_{21} یا p_{12}, p_{22}. معادله مستقل بعدی، معادله نرمال سازی است. اگر معادله اول Chapman – Kolmogorov را بکارگیریم:

$\pi_1 = \pi_1\ p_{11} + \pi_2\ p_{21} = \pi_1\ (1/m) + \pi_2\ (1/m)$

$\pi_2 = (m - 1)\ \pi_1$

$\pi_1 + \pi_2 = 1 = m\ \pi_1 \rightarrow \pi_1 = 1/m\ \&\ \pi_2 = (m - 1)/m$

همزمانی یا توازی متوسط ماژول های حافظه برابر است با:

$B = 1(1/m) + 2(m - 1)/m = (2m - 1)/m$

مثال ۲: ۳ پردازنده و $m \geq 3$ ماژول.

حالت	
۱	۳ درخواست در یک ماژول
۲	۲ درخواست در یک ماژول، ۱ درخواست در ماژول دیگر
۳	۱ درخواست در هر کدام از ۳ ماژول

$p_{11} = (1/m).1 = 1/m$

$p_{12} = (1/m).C(m-1, 1) = (1/m)(m-1)$

$p_{13} = 0$

$p_{21} = (1/m)^2.1 = (1/m)^2$

$p_{22} = (1/m)^2(C(m-1, 1) + C(m-1, 1)C(2,1)) = (1/m)^2 3(m-1)$

$p_{23} = (1/m)^2.C(m-1, 2)P(2, 2) = (1/m)^2 (m-1)(m-2)$

$p_{31} = (1/m)^3.C(m, 1) = (1/m)^3 m$

$p_{32} = (1/m)^3.C(m, 2)C(3, 2)P(2, 2) = (1/m)^3 3m(m-1)$

$p_{33} = (1/m)^3.C(m, 3)P(3, 3) = (1/m)^3 m(m-1)(m-2)$

از معادلات Chapman – Kolmogorov برای π_1 و π_3 به علاوه معادله نرمال سازی استفاده می کنیم.

$\pi_1 = \pi_1(1/m) + \pi_2(1/m^2) + \pi_3(1/m^2)$

$\pi_3 = \pi_2 (m-1)(m-2)/m^2 + \pi_3 (m-1)(m-2)/m^2$

$\pi_1 + \pi_2 + \pi_3 = 1$

$\rightarrow$

$\pi_1 = 1 / (m^2 - m + 1)$

$\pi_2 = (3m-2)(m-1) / (m(m^2 - m + 1))$

$\pi_3 = (m-1)^2(m-2) / (m(m^2 - m + 1))$

$\rightarrow B = 1(1/(m^2 - m + 1))$

$+ 2(3m-2)(m-1) / (m(m^2 - m + 1))$

$+ 3(m-1)^2(m-2) / (m(m^2 - m + 1))$

$= (3m^3 - 6m^2 + 6m - 2) / (m(m^2 - m + 1))$

مثال ۳: ۴ پردازنده، ۴ ماژول

حالت	
1	همه چهار درخواست در یک ماژول
2a	۳ درخواست در یک ماژول و یک درخواست در ماژول دیگر
2b	۲ درخواست در یک ماژول و ۲ درخواست در ماژول دیگر
3	۲ درخواست در یک ماژول و یک درخواست در دو ماژول دیگر
4	یک درخواست در هر کدام از چهار ماژول

نام حالت ها را طوری انتخاب کردیم که بیانگر تعداد ماژول های مشغول در هر حالت باشد.
احتمالات انتقال حالت اینگونه اند:

$p_{11} = 1/4$

$p_{12a} = 1/4\ C(3, 1) = 3/4$

$p_{12b} = p_{13} = p_{14} = 0$

$p_{2a1} = (1/4)^2$

$p_{2a2a} = (1/4)^2 C(3, 1)P(2, 2) = 6/16$

$p_{2a2b} = (1/4)^2\ C(3, 1) = 3/16$

$p_{2a3} = (1/4)^2\ C(3, 2)P(2, 2) = 6/16$

$p_{2a4} = 0$

$p_{2b1} = 0$

$p_{2b2a} = (1/4)^2\ C(2, 1) = 2/16$

$p_{2b2b} = (1/4)^2\ P(2, 2) = 2/16$

$p_{2b3} = (1/4)^2\ C(2, 1) + (1/4)^2\ C(2, 1)C(2, 1)P(2, 2) = 10/16$

$p_{2b4} = (1/4)^2\ P(2, 2)$

$p_{31} = (1/4)^3 = 1/64$

$p_{32a} = (1/4)^3\ C(3, 1) + (1/4)^3\ C(3, 1)C(3, 2) = 12/64$

$p_{32b} = (1/4)^3\ C(3, 1)C(3, 2) = 9/64$

$p_{33} = (1/4)^3\ C(3, 2)P(3, 3) + (1/4)^3\ C(3, 2)C(3, 2)P(2, 2) = 36/64$

$p_{34} = (1/4)^3\ P(3, 3) = 6/64$

$p_{41} = (1/4)^4\ C(4, 1) = 4/256$

$p_{42a} = (1/4)^4\ C(4, 2)C(4, 3)P(2, 2) = 48/256$

$p_{42b} = (1/4)^4\ C(4, 2)C(4, 2) = 36/256$

$p_{43} = (1/4)^4\ C(4, 3)C(3, 1)C(4, 2)P(2, 2) = 144/256$

$p_{44} = (1/4)^4\ P(4, 4) = 24/256$

$$\rightarrow \quad P = \begin{bmatrix} 1/4 & 3/4 & 0 & 0 & 0 \\ 1/16 & 6/16 & 3/16 & 6/16 & 0 \\ 0 & 2/16 & 2/16 & 10/16 & 2/16 \\ 1/64 & 12/64 & 9/64 & 36/64 & 6/64 \\ 4/256 & 48/256 & 36/256 & 144/256 & 24/256 \end{bmatrix}$$

$\boldsymbol{\pi P = \pi}$

$$\rightarrow \quad \begin{bmatrix} \pi_1 \\ \pi_{2a} \\ \pi_{2b} \\ \pi_3 \\ \pi_4 \end{bmatrix} = \begin{bmatrix} 0.0323 \\ 0.2419 \\ 0.1452 \\ 0.5081 \\ 0.0726 \end{bmatrix}$$

توازی متوسط ماژول ها برابر است با:

$B = \pi_1.1 + (\pi_{2a} + \pi_{2b}).2 + \pi_3.3 + \pi_4.4 = 2.2610$

با این سه مثال می توانیم توازی متوسط چهار ماژول ها را با دو، سه و چهار پردازنده مقایسه کنیم:

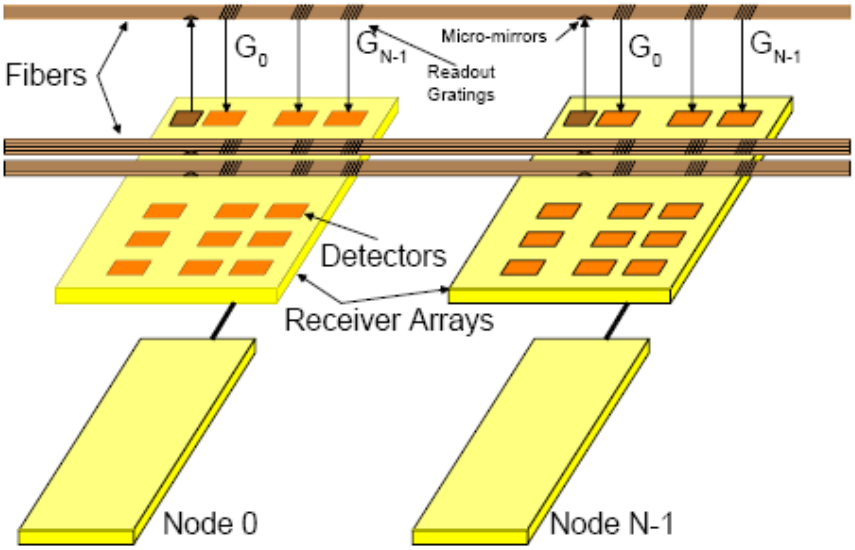
p	B
2	1.750
3	2.269
4	2.261

از رفتن دو به سه پردازنده، ۳۰% افزایش توازی بین ماژول های حافظه دیده می شود. وقتی از سه به چهار پردازنده می رویم، کارایی فقط حدود ۱۵% افزایش می یابد.

معماری SOME – Bus broadcast

مفیدترین ویژگی های شبکه میان ارتباطی پردازنده های موازی، پهنای باند بالا(مستقیما با تعداد پردازنده ها زیاد شود)، تأخیر کم، نبود تأخیر داوری[۲۴] و ارتباطات غیربلوکه شونده است. باس تعویض چندپردازنده نوری همزمان[۲۵] (SOME – BUS) [۱۱]، مستقیما هر پردازنده را به همه پردازنده ها وصل می کند و قادر به دربرگیری بیش از یکصد پردازنده بدون ایجاد گلوگاه[۲۶] است.

یکی از اصلی ترین ویژگی های آن، این است که هر گره، یک کانال broadcast اختصاصی دارد که با یک گروه از طول موج ها در یک فیبر خاص درست شده است و یک واسط ورودی دارد که از آرایه ای از گیرنده ها تشکیل شده است که بطور همزمان، تمام کانال ها را مانیتور می کند و درنتیجه یک شبکه قویا متصل برقرار می سازد. شکل ۱، آرایه گیرنده را نشان می دهد.



شکل ۱: آرایه گیرنده

یک طول موج جداگانه برای سیگنال کلاک در نظر گرفته شده است که برای همه جریان های داده در یک فیبر، مشترک است. یک پیاده سازی معمولی از این سیستم با ۱۲۸ گره و گیرنده های CMOS، تشخیص دهنده های Si غیرمتبلور، ۳۲ فیبر با ۱۷ طول موج (هر گره ۴ تا بعلاوه کلاک مشترک) و نرخ کلاک ۱۵۵ مگاهرتز می تواند پهنای باند ۷۷ مگابایت بر ثانیه را برای هر کانال فراهم آورد. با تکنولوژی پیشرفته تری مثل گالیوم – آرسنایدو نرخ کلاک ۱ گیگاهرتز برای طول موج، SOME – Bus ۳۲ فیبری مشابه می تواند پهنای باند ۵۰۰ مگابایت برثانیه را برای هر گره فراهم کند.

پیغام های جابجا شده بین گره ها، یک فیلد سرآمد[۲۷] دارند که اطلاعات نوع پیغام، طول پیغام و آدرس مقصد را در خود دارد. واسط ورودی کانال، کار فیلتر کردن آدرس، پردازش مانع[۲۸]، نشان دادن طول[۲۹] و رمزگشایی نوع پیغام را انجام می دهد و می تواند بطور همزمان، از هر تعداد گره، پیغام دریافت کند و آنها را در یک صف نگه دارد تا زمانیکه پردازنده محلی بتواند آنها را از صف خارج سازد. هر کانال ورودی یک صف دارد که اجازه می دهد تعداد دلخواهی از پیغام ها بطور همزمان برسند. پیغام های همزمان سازی، جمع آوری می شوند و در گیرنده پردازش می گردند. علاوه بر تشخیص آدرس خودش، گیرنده می تواند آدرس گروه های multicast, broadcast را نیز تشخیص دهد.

SOME – Bus ممکن است مثل crossbar به نظر آید ولی عملکرد بهتری نسبت به آن دارد. یکی از مزایای این معماری این است که بخاطر قابلیت broadcast چندگانه، هیچ گره ای توسط فرستنده دیگری، موقع فرستادن، بلوکه نمی شود، هیچ داوری مورد نیاز نیست و پهنای باند شبکه مستقیما با تعداد گره ها، افزایش می یابد. اگر N گره داشته باشیم، قطر SOME – Bus برابر ۱ است، زمان ارتباطات همه – به – همه[۳۰] با پیغام های متفاوت، O(N) است و زمان همزمان سازی، O(1) است. برخلاف یک شبکه قویا متصل، که تعداد فرستنده ها و کانال ها، $O(N^2)$ است، تعداد فرستنده ها و کانالهای SOME – Bus، O(N) است که از تعداد مورد نیاز در معماری های دیگر مثل ابر مکعب[۳۱] و توروس[۳۲] کمتر است. تعداد گیرنده ها N^2 است که از تعداد مورد نیاز در معماری های دیگر بیشتر است. آنها طوری چیده شده اند که N گیرنده بصورت ساختارهای سیلیکون غیرمتبلور روی یک فیلم نازک و مستقیما روی سطح یک CMOS دیجیتال ساخته می شوند و به لیتوگرافی احتیاجی ندارند. بخاطر رسانایی کم لایه سیلیکون غیرمتبلور، هیچ الگوسازی[۳۳] لازم نیست و در نتیجه، بازده[۳۴] و قیمت گیرنده توسط بازده و قیمت دستگاه CMOS تعیین می گردد. چون گیرنده به هیچ مسیریابی نیاز ندارد، پیچیدگی سخت افزاری آن و در نتیجه قیمت آن کم است. کل آرایه گیرنده، حتی برای مقادیر بزرگ N ($N > 128$) می تواند روی یک تراشه ساخته شود. در نتیجه، قیمت کل گیرنده تقریبا O(N) است بجای اینکه $O(N^2)$ باشد.

SOME – Bus می تواند یک سیستم CC – NUMA را پشتیبانی کند که در آن فضای آدرس مجازی مشترک، بین حافظه های محلی توزیع شده است و می تواند هم توسط پردازنده محلی و هم پردازنده های دور مورد دسترسی قرار گیرد البته با تأخیرهای متفاوت. ترافیک روی شبکه میان ارتباطی از پیغام های داده تشکیل می گردد که بخاطر عدم برخورد کش روی یک گره (محلی) و رفتن به حافظه یک گره دیگر (دور) ایجاد می شوند و همچنین از پیغام های دیگری تشکیل می گردد که بخاطر حفظ سازگاری کش ها رد و بدل می شوند. با اینکه SOME – Bus می تواند از تکنیک های نرم افزاری برای پیاده سازی سازگاری کش استفاده کند، ولی می تواند از سخت افزار فرستنده، گیرنده و کنترلر کش برای ایجاد یک مکانیزم سازگاری کش در سطح سیستم استفاده نماید. این سیستم مبتنی بر سخت افزار، از بلوک های کش برای انجام سازگاری استفاده می نماید که احتمال اشتراک غلط و خرابی سیستم را در مقایسه با سیستم های نرم افزاری که از بلوک های بزرگتری مثل صفحه های حافظه استفاده می نمایند، کاهش می دهد.

تجسس[۳۵] یک تکنیک رایج برای حفظ سازگاری است. این تکنیک نیاز دارد تا تمام کش ها، هر درخواست نوشتن در حافظه از هر پردازنده را ببینند و در گذشته، مقیاس پذیری سیستم های DSM را محدود ساخته بود چون که شبکه میان ارتباطی، حتی با تعداد کمی پردازنده، سریعا اشباع می شد. SOME – Bus این مشکل را ندارد. هر پردازنده، می تواند از طریق کانال خودش، به سادگی، پیغام های به روز رسانی[۳۶] یا نامعتبرسازی[۳۷] را broadcast کند. هر گیرنده نیز می تواند کانال های ورودی خود را برای پیغام های نامعتبرسازی ببیند و به کنترلر کش بگوید که چه کاری بکند. با اینکه امکان اشباع شبکه میان ارتباطی از بین رفته است، ترافیک شدید سازگاری کش، می تواند کنترلر کش را اشباع سازد. یک سیستم مبتنی بر SOME – Bus می تواند از مزایای تکنیک های مبتنی بر دایرکتوری استفاده کند که فقط آن کش هایی را مطلع می سازد که بلوک های آنها مورد تغییر قرار گرفته اند. این امر می تواند به سادگی با داشتن یک لیست از مقصد ها در header پیغام نامعتبرسازی انجام شود. پیغام ها هنوز از طریق کانال خروجی گره فرستنده، broadcast می گردند ولی تصمیم پذیرش یا رد یک پیغام ورودی، بجای اینکه در کنترلر کش هر گره انجام پذیرد، در ورودی گیرنده انجام می شود.

آرایه گیرنده که در روی یک تراشه قرار گرفته است، می تواند در باس پردازنده – حافظه ظاهر شود تا پردازنده بتواند بعنوان یک قسمت از حافظه به آن دسترسی یابد. علاوه بر این، آرایه گیرنده، برای پیاده سازی کارای پروتکل، به کش و کنترلر دایرکتوری متصل شده است.

مدل های حافظه مشترک توزیع شده

<u>مدل ۱</u>

در یک سیستم SOME – Bus با N گره، هر گره، یک پردازنده با کش، حافظه، یک کانال خروجی و یک گیرنده دارد که می تواند پیغام ها را بطور همزمان از همه N کانال دریافت کند. برای پشتیبانی پروتکل MSI، هر گره، یک دایرکتوری نیز دارد که اطلاعات سازگاری را در مورد آن بخش از حافظه توزیع شده که در آن گره پیاده سازی شده است، نگه می دارد. یک مدل اجرای چند ریسمانی[۳۸] فرض می شود: هر پردازنده، برنامه ای را اجرا

می کند که شامل M ریسمان[۳۹] موازی است. گره ای که M ریسمان در آن اجرا می گردد، گره میزبان[۴۰] یا مالک[۴۱] نامیده می شود. ریسمان ها فقط می توانند در گره مالک خود اجرا گردند. یک ریسمان آنقدر به اجرای خود ادامه می دهد تا با یک عدم برخورد کش سراسری مواجه شود که به یک داده یا اجازه از یک گره دور نیازمند گردد. سپس معلق می ماند تا عمل درخواست شده به اتمام برسد(داده از یک گره دور انتقال یابد یا اجازه دریافت گردد) و در این موقع، برای اجرا آماده می گردد و نهایتا اجرای خود را از سر می گیرد. یک ریسمان در یک پردازنده برای مدت زمان R اجرا می گردد تا به حالت تعلیق برود. یک عدم برخورد در کش باعث می شود تا یک پیغام درخواست در صف کانال خروجی قرار گیرد. بعد از اتمام زمان انتقال T، پیغام در صف ورودی گیرنده گره مقصد(دور) قرار می گیرد، توسط دایرکتوری، سرویس می شود و یک پیغام دیگر به گره اول برمی گردد که داده یا وصول[۴۲] را در خود دارد. حافظه گره دور، به زمان L برای ساختن پیغام پاسخ نیاز دارد. زمان L توسط کنترلر دایرکتوری صرف می گردد تا دسترسی های لازمه به حافظه، ساختن پیغام پاسخ و قراردادن آن در صف کانال خروجی گره دور انجام گردد. بعنوان قسمتی از سرویس دادن به یک پیغام، دایرکتوری ممکن است به دیگر گره ها پیغام هایی بفرستد و داده یا وصول را دریافت کند. یک عدم برخورد کش سراسری می تواند بعلت یک عدم برخورد خواندن[۴۳] یا نوشتن[۴۴] در کش محلی باشد. به تناسب، یک درخواست داده[۴۵] یا درخواست مالکیت[۴۶] به دایرکتوری گره مالک فرستاده می شود. وقتی این پیغام دریافت گردید، بلوک داده مورد نظر می تواند در حالت مشترک[۴۷]یا تغییریافته[۴۸] باشد. دایرکتوری مالک ممکن است یک پیغام داده به درخواست کننده بفرستد(موقع خواند یک بلوک مشترک) یا ممکن است پیغام نامعتبرسازی بفرستد و پیغام های وصول آنرا دریافت کند.

جدول ۱: لیست نمادها

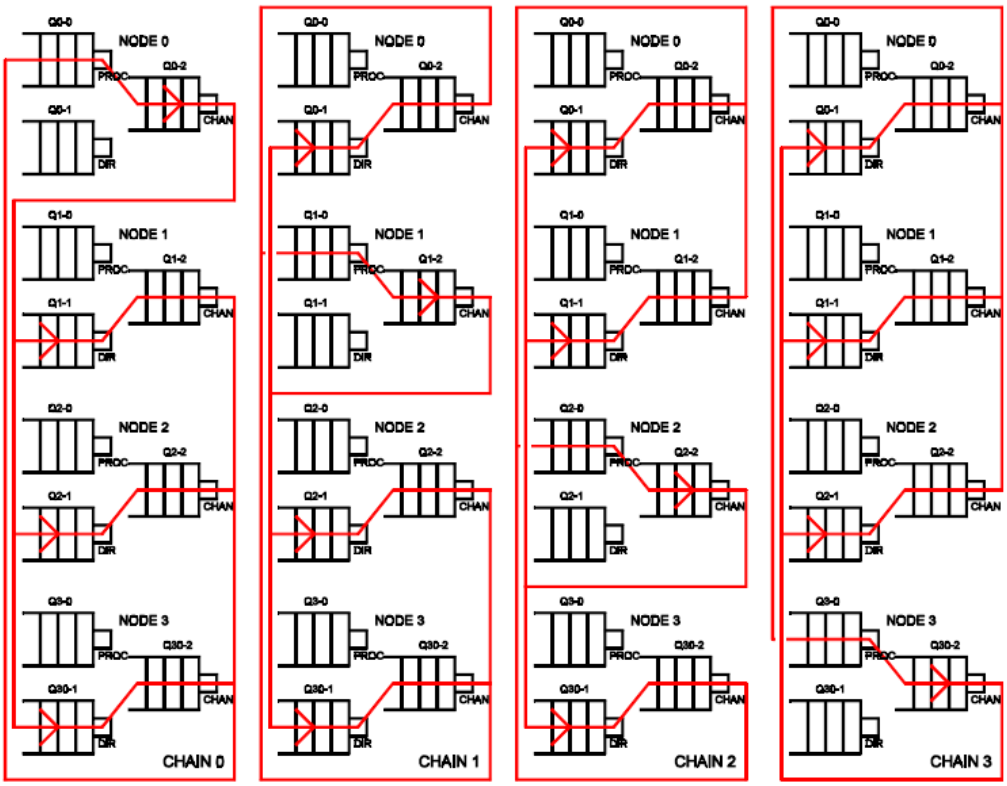
K_D	تعداد کل پیغام های داده در سیستم
K_C	تعداد متوسط پیغام های سازگاری در سیستم
λ_D	نرخ ورود درخواست های سازگاری
L	زمان کنترلر دایرکتوری
L_1	زمان ساختن یک پیغام وصول – داده
L_2	زمان متوسط انتقال کانال

L_3	زمان متوسط صف کانال
L4	زمان متوسط فرستادن پیغام های نامعتبرسازی و دریافت وصول نامعتبرسازی
L5	زمان متوسط دریافت وصول نامعتبرسازی
M	تعداد ریسمان های موازی
N	تعداد گره ها
N_{inv}	تعداد بلوک هایی که در موقع درخواست مالکیت، نامعتبر می شوند
p_{rd}	احتمال اینکه پیغام داده بخاطر عدم برخورد خواند باشد
p_{sh}	احتمال اینکه یک بلوک در حالت مشترک باشد
R	زمان متوسط ریسمان
R_D	زمان متوسط رفت و برگشت پیغام داده
T	زمان متوسط انتقال کانال پیغام

از آنجا که M نشان دهنده تعداد ماکزیمم درخواست های معلقی است که یک گره قبل از بلوکه شدن می تواند داشته باشد، فرض می شود که وقتی یک گره، کمتر از M درخواست معلق دارد، با زمان متوسط R ، درخواست تولید می کند. R، زمان متوسط ریسمان است. یک پیغام درخواست تولید شده توسط یک گره، با احتمال مساوی به بقیه گره ها فرستاده می شود. این نوع مفروضات برای عملکرد گره ها، مثل مفروضاتی هستند که در [۱۲] صورت گرفته است و مثل آنهایی هستند که در [۴] و [۱۳] برای مطالعه کارایی DSM در یک سیستم توروس استفاده شده اند. چون یک پیغام توسط یک گره به گره مالک فرستاده می شود، که گره مالک به نوبه خود با یک پیغام دیگر به آن پاسخ می دهد، این نوع عملیات را می توان با یک شبکه صف بسته چند زنجیره ای[۴۹] نشان داد که پیغام ها، نوع پیچیده تری از سرویس را در سرورهای خاصی دریافت می دارند. M ریسمان درون هر گره، یک کلاس جداگانه از پیغام ها را تشکیل می دهند. وقتی $m < M$ ریسمان، پیغام های معلق دارند، $M - m$ پیغام باقیمانده، بصورت FCFS در پردازنده سرویس می شوند. یک پیغام در پردازنده مالک سرویس می شود(با توزیع هندسی با متوسط R)، در صف کانال خروجی پردازنده مالک قرار می گیرد، توسط کانال (در زمان T) سرویس می شود، در صف ورودی گیرنده گره دور

قرار می گیرد، توسط دایرکتوری گره دور(در زمان L) سرویس می شود، و بطور مشابه از طریق کانال خروجی گره دور و صف ورودی گیرنده گره مالک، به گره مالک برمی گردد.

بخاطر تقارن سیستم، تمام کنترلرهای دایرکتوری به یک صورت عمل می کنند. بنابراین، یک سیستم با N=M+1 گره، کارایی یک سیستم با N > M + 1 گره را دارد. معمولا تعداد کمی ریسمان در هر گره امکان پذیر است. آنگاه یک شبکه صف نسبتا کوچک، که یک سیستم SOME – Bus با M+1 گره را نشان می دهد، می تواند برای اندازه گیری معیارهای کارایی بکار رود. گره P (P = 0, 1, …, M) دارای M پیغام داده معمولی است که در میان پردازنده، کنترلرهای دایرکتوری دیگر گره ها و کانال های مربوطه در گردش هستند. این پیغام ها نمایانگر پیغام های درخواست – داده و درخواست – مالکیت از یک گره دور به گره مالک و پیغام های برگردانده شده وصول – داده[۵۰] و وصول – مالکیت[۵۱] به گره اولیه می باشند. ما از زنجیره های متفاوت برای تمییز دادن پیغام های متعلق به پردازنده های متفاوت استفاده می کنیم بطوری که پیغام های درون زنجیره P به گره P تعلق دارند. شکل ۲ شبکه صف یک سیستم با N = 4 گره را نشان می دهد(M=3). پیچیدگی مدل از این واقعیت ناشی می شود که پیغام های گوناگون دیگری وجود دارند که فقط برای حفظ سازگاری به کار می روند. این پیغام ها از درون پردازنده ها نمی گذرند. به جای آن، آنها توسط کنترلرهای دایرکتوری ایجاد می شوند، از کانال ها می گذرند، احتمالا با کنترلرهای کش سروکار دارند و نهایتا به کنترلرهای دایرکتوری اولیه برمی گردند. این ترافیک اضافی سازگاری، دو اثر مستقیم بر روی پیغام های داده معمولی زنجیره ها خواهد داشت.



شکل ۲: مدل ۱، شبکه صف (۴ گره)

اول اینکه ترافیک سازگاری، زمان سرویس در کنترلر دایرکتوری را تعیین می کند. ما فرض می کنیم که پیغام های داده، با احتمال p_{rd} بخاطر عدم برخورد خواندن می باشند و یک بلوک با احتمال p_{sh} در حالت مشترک به سر می برد. چهار حالت مجزا وجود خواهد داشت که در هر کدام، یک عمل متفاوت توسط کنترلر دایرکتوری در گره مالک انجام می گردد.

الف) درخواست داده به یک بلوک در حالت مشترک(احتمال $p_{rd}*p_{sh}$). کنترلر دایرکتوری، پیغام درخواست داده را دور می اندازد و پیغام وصول – داده را به گره درخواست کننده برمی گرداند. از نقطه نظر شبکه صف، بین یک پیغام درخواست – داده و وصول – داده تفاوتی موجود نیست. این عملیات به سادگی می تواند بصورت یک پیغام درخواست – داده دیده شود که سفری را از پردازنده خود به کنترلر دایرکتوری مالک انجام داده و به پردازنده خود بازمی گردد. زمان سرویس آن در کنترلر دایرکتوری مالک، زمان L_1 است که دایرکتوری نیاز دارد تا پیغام وصول – داده را با یک کپی از بلوک درخواست شده بسازد.

ب) درخواست – داده به یک بلوک در حالت تغییریافته(احتمال $(p_{rd}*(1-p_{sh})$). یک پیغام نامعتبرسازی توسط دایرکتوری مبدا به گره ای که مالکیت بلوک را دارد فرستاده می شود. آن گره، یک پیغام وصول – نامعتبرسازی (همراه با پس نویسی[۵۲]) را به گره مبدا و گره درخواست کننده broadcast می کند.

ج) درخواست – مالکیت به یک بلوک در حالت تغییر یافته (احتمال $(1-p_{rd}))*(1-p_{sh})$). یک پیغام نامعتبرسازی توسط دایرکتوری مبدا به گره ای که مالکیت بلوک را دارد فرستاده می شود. آن گره یک پیغام وصول – نامعتبر سازی(همراه با پس نویسی) به گره مبدا و گره درخواست کننده می فرستد.

د) درخواست – مالکیت به یک بلوک در حالت مشترک (احتمال $(1-p_{rd}))*p_{sh}$). کنترلر دایرکتوری مبدا پیغام های نامعتبرسازی را به همه گره هایی که یک نسخه از بلوک درخواست شده را دارند، broadcast می کند. تمام پیغام های وصول – نامعتبرسازی را جمع آوری کرده و سپس یک پیغام وصول – مالکیت را به گره درخواست کننده می فرستد.

لذا زمان متوسط سرویس یک پیغام داده در کنترلر دایرکتوری مبدا، بدین صورت محاسبه می شود:

$$L = L_1.p_{rd}.p_{sh} + (L_2 + L_3).(p_{rd}.(1 - p_{sh}) + (1 - p_{rd}).(1 - p_{sh})) + L_4.(1 - p_{rd}).p_{sh} \qquad (1)$$

که L_1 زمانی است که دایرکتوری نیاز دارد تا پیغام وصول – داده را با یک کپی از بلوک درخواست شده، بسازد . L_2 زمان انتقال کانال است، L_3 زمان متوسط صف کانال است و $L_4 = (L_2+L_3)+L_5$، زمان متوسط فرستادن نامعتبرسازی ها و جمع آوری وصول های نامعتبرسازی است. فرض می کنیم که کنترلر کش با سرعت بالاتری نسبت به شبکه کارمی کند لذا از زمان پاسخ کنترلر کش می توان چشم پوشی نمود. اگر تعداد بلوک هایی که باید در هر درخواست – مالکیت نامعتبر شوند، مقدار ثابت N_{inv} باشد، آنگاه L_5 مقدار متوسط یک متغیر تصادفی برابر با مقدار ماکزیمم N_{inv} متغیر تصادفی یکسان می باشد که هر کدام برابر زمان سرویس و صف در سرور کانال می باشند. در کاربردهای گوناگون، مشاهده شده است که معمولا یک پیغام نامعتبرسازی فرستاده می شود. در این حالت، $L_5 = L_2+L_3$ و

$L_4 = 2.(L_2+L_3)$ است.

دوم اینکه ترافیک سازگازی نیز از کانال ها می گذرد. تداخل با پیغام های داده می تواند با این واقعیت تقریب زده شود که پیغام های سازگاری، قسمتی از نرخ سرویس سرور کانال را می بلعند و زمان سرویس کانال برای پیغام های داده، بیشتر می شود. این زمان سرویس برابر است با:

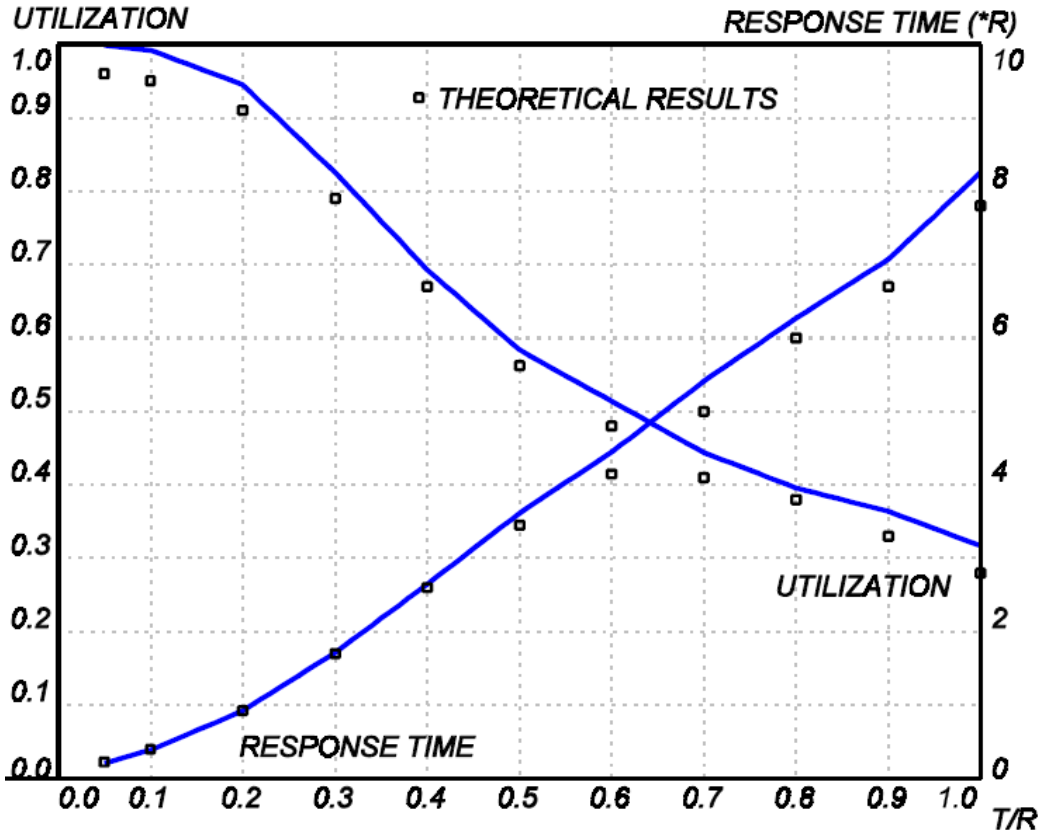
$$T' = T.(K_D+K_C)/K_D \qquad (2)$$

که T، زمان متوسط واقعی انتقال کانال است، $K_D = M.(M+1)$، تعداد کل پیغام های داده در سیستم است و K_C تعداد متوسط پیغام های سازگاری موجود در سیستم می باشد. برای محاسبه K_C، باید توجه کنیم که پیغام های سازگاری بدلیل ورود پیغام های داده درون مکانیزم سازگاری تولید می شوند. آنگاه از فرمول Little استفاده می کنیم: $K_C = \lambda_D.L_4$ که λ_D نرخ ورود پیغام های داده درون مکانیزم سازگاری و L_4 زمانی است که یک پیغام سازگاری در سیستم موجود می باشد. نرخ ورود λ_D برابر $\lambda_D = (1-p_{rd}.p_{sh}).K_D/R_D$ که R_D زمان متوسط رفت و برگشت پیغام داده می باشد و ضریب $(1-p_{rd}.p_{sh})$ ضروری است چون پیغام های درخواست – داده به یک بلوک در حالت مشترک، باعث ترافیک سازگاری اضافه نمی گردند.

با استفاده از نرخ سرویس کاهش یافته کانال و تخمین زمان سرویس دایرکتوری، شبکه صف بسته با $3(M+1)$ صف، $(M+1)$ زنجیره و M پیغام در هر زنجیره را می توان با تکنیک های استاندارد حل نمود. تنها پارامتر نامعلوم در معادلات بالا، زمان صف کانال است. از یک مقدار اولیه می توان استفاده نمود (مثلا برابر زمان سرویس کانال) و مدل را بصورت تکراری[۵۳] حل نمود. جواب همگرا می شود چون یک مقدار نامناسب برای زمان صف کانال، به یک مقدار نامناسب برای زمان سرویس دایرکتوری منجر می شود که باعث می گردد مدل، یک مقدار جدید برای زمان صف کانال تولید کند که باز هم نامناسب است ولی تأثیر برعکس خواهد داشت(یعنی یک مقدار اولیه که خیلی کوچک باشد، باعث یک مقدار جدید برای زمان صف کانال می گردد که خیلی بزرگ است).

با این فرض که همه زمان های پردازش و انتقال دارای توزیع هندسی هستند، شکل ۳ کارایی SOME – Bus را که بصورت بازدهی پردازنده و زمان پاسخ اندازه گیری شده

است، نشان می دهد. زمان پاسخ، زمان بین موقعی است که یک عدم برخورد کش باعث می شود تا یک پیغام در صف کانال خروجی قرار گیرد تا موقعی که پیغام داده (یا وصول) متناظر وارد صف ورودی گردد. در شکل ۳، احتمال اینکه یک پیغام، یک پیغام داده باشد(بخاطر عدم برخورد خواندن) برابر $p_{rd} = 0.8$ است؛ احتمال اینکه یک درخواست – مالکیت باشد(بخاطر عدم برخورد سراسری کش) 0.2 است. این مفروضات با الگوهای معمول مراجعه به حافظه سازگار است که سایکل های نوشتن، حدود ۱۵% تا ۲۰% کل سایکل های حافظه را تشکیل می دهند. احتمال اینکه یک بلوک در حالت مشترک باشد برابر است با $p_{sh} = 0.8$. نسبت زمان متوسط انتقال به زمان متوسط اجرای ریسمان بین ۰,۰۵ و ۱ متغیر است. این بازه برای مدل کردن رفتار سیستم تحت تنظیمات و رفتار کش رایج کافی می باشد. فرض کنید m نرخ عدم برخورد و F تعداد دستورالعمل بر ثانیه باشد که توسط پردازنده در هر گره انجام می شود. همچنین فرض کنید S اندازه متوسط پیغام به بایت باشد و C پهنای باند کانال باشد(به بایت بر ثانیه). آنگاه زمان متوسط اجرای ریسمان برابر $R = 1/(mF)$ و زمان متوسط انتقال پیغام برابر $T = S/C$ است. نسبت این دو $T/R = mSF/C$ می باشد. در معماری های با کارایی بالای امروزی، نسبت F/C در بازه ۰,۵ تا ۱ می باشد. مثلا در Cray T3D، $F = 150 \cdot 10^6$ و لینک های شبکه در 150MHz کار می کنند. در Cray T3E، لینک های شبکه ۲ تا ۴ برابر پهنای باند بیشتری دارند؛ در ASCI RED، هر گره دو پردازنده دارد با $F = 200.10^6$ و لینک های شبکه 400MB/S در هر جهت می باشند. با اندازه کوچک برای بلوک های کش و نرخ عدم برخورد در حدود ۱۰% یا کمتر، نسبت T/R در محدوده ۰,۰۵ تا ۱ قرار می گیرد.



شکل ۳: مدل ۱، کارایی SOME – Bus

نتایج شبیه سازی و مقایسه

این بخش نتایج شبیه سازی را برای مقایسه کارایی معماری SOME – Bus با کارایی یک توروس دو بعدی و معماری circuit – switched crossbar با یک مدل شبکه صف ارائه می دهد.

در معماری توروس، هر گره به چهار همسایه خود متصل است. یک گره، چهار صف خروجی دارد که پیغام ها را از کانال های مربوطه به گره های مقصد می رسانند. مسیریابی wormhole با یک پروسه تطبیقی برای انتخاب کانال خالی بعدی استفاده شده است. در معماری crossbar، هر گره، یک کانال خروجی و یک کانال ورودی دارد. Crossbar می تواند کانال خروجی یک گره را به کانال ورودی یک گره دیگر متصل سازد. یک پیغام در صف کانال خروجی منتظر می ماند، اگر که کانال ورودی مقصد توسط یک کانال خروجی دیگر استفاده شده باشد. وقتی یک گره بخواهد یک پیغام مشابه را multicast کند، آنقدر صبر

مـی کنـد تـا همـه کانـال هـای ورودی مـورد نیـاز آزاد گردنـد، آنهـا را رزرو مـی کنـد و بطـور همزمان یک کپی از پیغام را بـه همـه گره ها مـی فرستد. انتخـاب گره مقصـد، بـرای همـه گره های SOME – Bus و crossbar و برای هر جهت[۵۴] در توروس، یکنواخت است.

یـک پیغـام نامعتبرسـازی در معمـاری SOME – Bus، broadcast مـی شـود. در معماری crossbar، یک گره آنقدر صبر مـی کند تـا تمـام کانـال هـای ورودی مـورد نیـاز آزاد گردند، آنها را رزرو می کند و یک پیغام نامعتبرسازی به همه گره هـای مقصـد مـی فرستد. در معمـاری تـوروس، یـک درخـت پوشـا از گـره مبـدا بـه همـه گـره هـای مقصـد ایجـاد مـی گـردد و wormhole های چند مقصده، پیغام های نامعتبرسازی را broadcast می کنند.

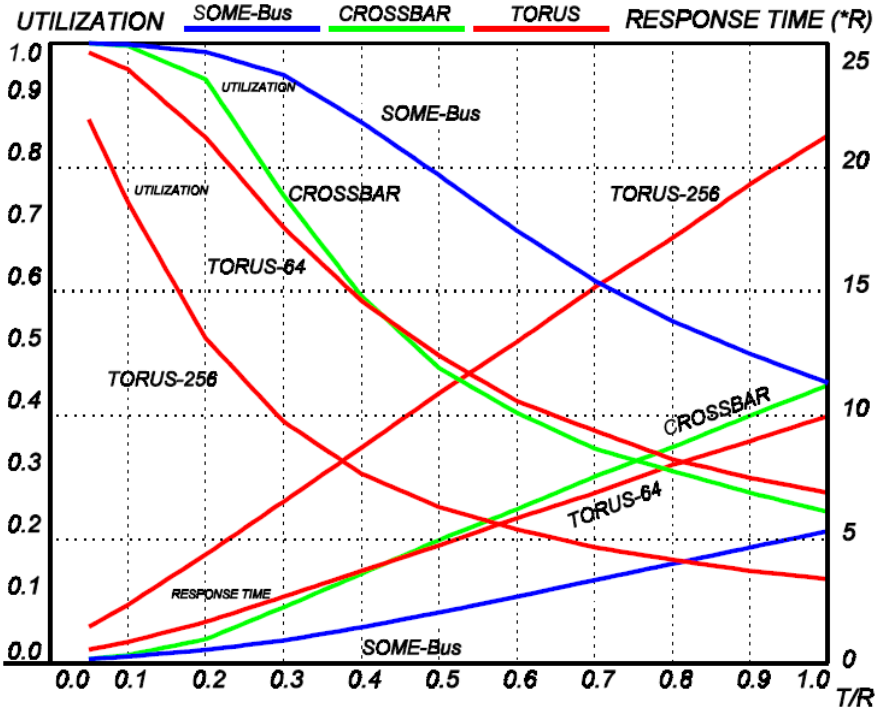
مهمترین پارامترهـای شبیه سـازی، توزیـع زمـان اجـرای ریسـمان و زمـان دایرکتـوری بـرای سـاختن یـک پیغـام پاسـخ، توزیـع انـدازه پیغـام هـا و توزیـع انتخـاب گـره مقصـد مـی باشـد. پارامترهای اضافی شبیه سازی، نسبت پیغام های نوشتن، تعداد پیغـام هـای نامعتبرسـازی کـه بـا هر پیغـام درخواسـت – مالکیـت فرسـتاده مـی شـوند و میـزان زمـان سرویسـی اسـت کـه در گـره مقصد دریافت می کنند. کمیت های زیادی برای اندازه گیری و مقایسه کارایی معمـاری استفاده می شود. اصلی ترین آنها، میانگین نسبت زمـان اجـرای ریسـمان هـا و زمـان متوسـط پاسـخ مـی باشد.

نقطه مرجع برای پارامترهای زمانی و ارزیابی شبیه سـازی، زمـان اجـرای ریسـمان R می باشد که فرض می شود بطور هندسی با میانگین ۱۰۰ واحد زمانی توزیع شده است. انـدازه پیغـام طـوری انتخـاب مـی گـردد کـه زمـان انتقـال T در محـدوده ۵ تـا ۱۰۰ واحـد زمـانی قـرار گیـرد(یعنـی نسـبت T/R در محـدوده ۰،۰۵ تـا ۱ قـرار گیـرد). در معمـاری تـوروس، T زمـان واقعی انتقال است فقط اگر wormhole در هیچ گره ای بین مبدا و مقصـد، بلوکـه نگـردد. در تمـام معمـاری هـا، پردازنـده هر گـره، یـک برنامـه بـا M = 3 ریسـمان را اجـرا مـی نمایـد. یـک درخواست – مالکیت به یـک بلـوک مشـترک منجـر بـه سـه نامعتبرسـازی مـی شـود. نتـایج بـرای سیستم هایی با ۶۴ یا ۲۵۶ گره مقایسه می شوند.

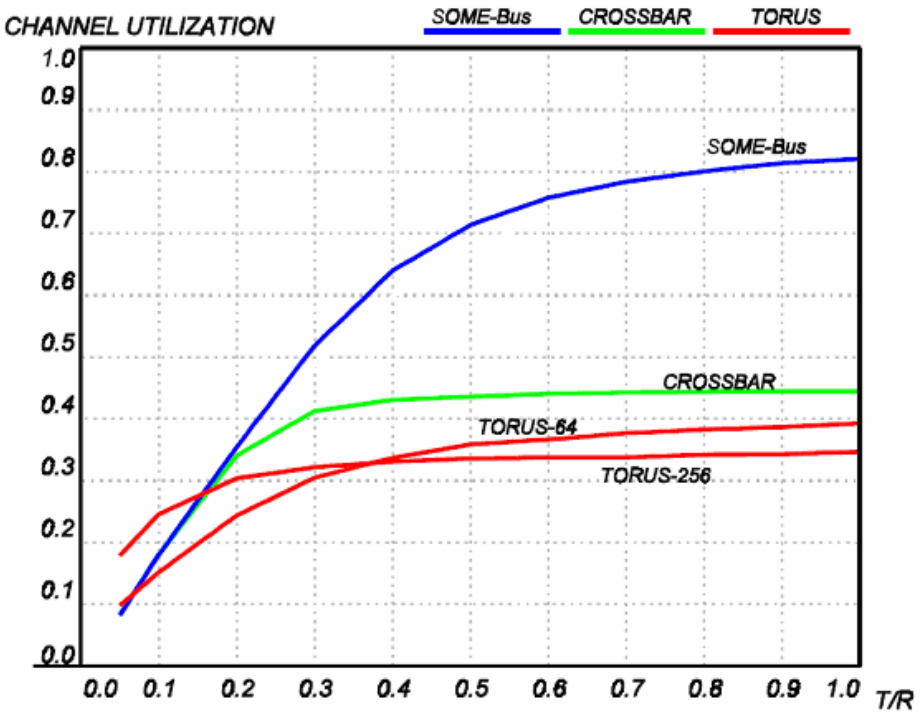
در شکل ۴، بازدهی پردازنده (یعنی نسبت زمانی که پردازنده مشـغول اجـرای ریسـمان های خودش است) برای سه معماری مقایسـه شـده اسـت و نسـبت زمـان متوسـط انتقـال بـه زمـان

متوسط اجرا بین ۰,۰۵ و ۱ تغییر می کند. همچنین فرض می کنیم که زمان انتقال پیغام، بجای اینکه توزیع هندسی داشته باشد، ثابت است. بخاطر کاهش واریانس، بازدهی پردازنده افزایش کمی را در مقایسه با بازدهی شکل ۳ از خود نشان می دهد.(بطور مشابه، زمان پاسخ بخاطر کاهش واریانس، کاهش می یابد). هر چقدر شدت ترافیک افزایش می یابد، بازدهی افت می کند و در توروس و crossbar محدود می شود. دلیل این امر افزایش عمده زمان انتظار در صف و افزایش زمان پاسخ، آنگونه که شکل ۴ نشان می دهد است. نتایج شبیه سازی روی سیستم توروس، به نظر می رسد که با نتایج تئوری در [۵] همخوانی داشته باشد. این نتایج شبیه سازی به این خاطر مهمند که وقتی نسبت T/R بزرگ می شود، کارایی قابل قبولی در معماری SOME – Bus بخصوص در مقایسه با معماری های دیگر را نشان می دهند. چنین T/R های بزرگی بخاطر نرخ عدم برخورد حدودا ۱۰% را می توان در بسیاری از کاربردها انتظار داشت. بازدهی در همه سیستم ها افت می کند ولی در معماری های توروس و crossbar سریعتر افت می کند که در مقادیر T/R بالای ۰,۴۵، بازدهی توروس و crossbar زیر ۵۰% می ماند. در یک سیستم با ۲۵۶ گره، کارایی معماری SOME – Bus و crossbar یکسان می ماند درحالیکه در توروس کاهش می یابد.

توانایی نسبی سه شبکه در ارسال بدون ازدحام پیغام ها را می توان با میانگین نسبت زمانی که کانال ها بیکار هستند توصیف نمود. همانگونه که شکل ۵ نشان می دهد، در معماری توروس، حتی وقتی شدت ترافیک افزایش می یابد، کانال ها بیشتر از ۵۰% اوقات بیکار می مانند ولی نمی توانند مورد استفاده قرار گیرند. محدودیت مشابهی در crossbar مشاهده شده است در حالی که در معماری SOME – Bus بازده کانال افزایش می یابد حتی وقتی که شدت ترافیک زیاد می گردد. یک علت چنین پدیده ای این است که وقتی کانال های مورد نیاز توسط دیگر wormholeها اشغال شده باشد، wormhole ها در توروس بلوکه می شوند. شبیه سازی نشان می دهد که تعداد متوسط دفعاتی که یک wormhole بلوکه می شود برای مدت زمان قابل ملاحظه ای بین ۱ و ۲ در سیستم ۶۴ گره ای و بین ۳و ۴ در سیستم ۲۵۶ گره ای می باشد. حتی برای مقادیر کوچک T/R، واضح است که wormhole با بلوکه شدن زیادی در همه زمان ها مواجه می شود و حتی تحت ترافیک متوسط، مسیریابی wormhole به مسیریابی store & forward تبدیل می گردد.



شکل ۴: مدل ۱، بازدهی پردازنده و زمان پاسخ



شکل ۵: مدل ۱، بازدهی کانال

مدل ۲

یکی از فرضیات رایج در اغلب مدل های DSM موجود در مقالات این است که رفتار پردازنده (و کش) تحت تأثیر فعالیت های کنترلر دایرکتوری قرار نمی گیرد(بغیر از کارهای مربوط به پروتکل). با این حال، کنترلر دایرکتوری، برای سرویس دادن به درخواست های وارده، به حافظه دسترسی پیدا می کند و متعاقبا، مقداری از پهنای باند حافظه را می گیرد که در حالت عادی می توانست در اختیار کنترلر کش محلی باشد. هر چقدر عدم برخورد کش بیشتر می شود و نرخ ورود پیغام در صف کنترلر دایرکتوری افزایش می یابد، کاهش بوجود آمده در پهنای باند کنترلر کش کمتر قابل اغماض می شود. در حقیقت، اگر گره بصورت یک چند پردازنده متقارن کوچک پیاده سازی شود، این پدیده حتی در نرخ های کم عدم برخورد هم می تواند رخ دهد.

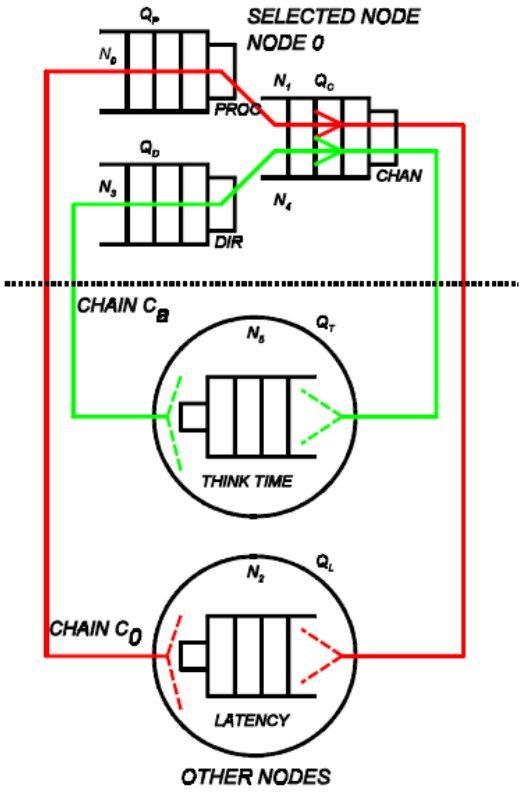
فرض اضافه ای که مدل ۲ انجام می دهد این است که موقعی که کنترلر دایرکتوری در حال سرویس دادن به یک درخواست است، با کنترلر کش که درحال تولید درخواست به حافظه است، به رقابت می پردازد. با مدل ۲، ما تداخلی را که یک پیغام موقع سرویس شدن در یک گره دور ایجاد می کند، مدل می کنیم. شبکه صف، همانند شبکه نشان داده شده در شکل ۲ است ولی تعاملی مبتنی بر سایکل – حافظه بین پردازنده و دایرکتوری وجود دارد. منبع مشترک، حافظه است: وقتی فقط یکی از دو کنترلر بخواهد به حافظه دسترسی پیدا کند، تمام پهنای باند را در اختیار خود می گیرد. وقتی هر دو کنترلر بخواهند دسترسی پیدا کنند، هرکدام قسمتی از پهنای باند را دریافت می سازد. این مد عملیاتی نیز با دو صف PROC, DIR در شکل ۲ مدل شده است. وقتی یک صف پیغام دارد و دیگری خالی است، پیغام موجود در ابتدای صف، سرویس کامل دریافت می دارد. وقتی هر دو صف پیغام دارند، دو پیغام موجود در ابتدای صف ها (و فقط آن دو)، با نسبت های α_1, α_2 (که $\alpha_1 + \alpha_2 = 1$) از ظرفیت سرور استفاده می کنند. زمان متوسط سرویس برای پیغام های دو صف، متفاوت است.

همانگونه که قبلا اشاره شد، بخاطر تقارن این معماری، سیستمی با $N = M + 1$ گره، کارایی یکسانی با سیستمی با $N > M + 1$ گره از خود نشان می دهد که M تعداد ریسمان های موجود در یک پردازنده است. معمولا تعداد کمی از ریسمانها در هر گره امکان پذیر

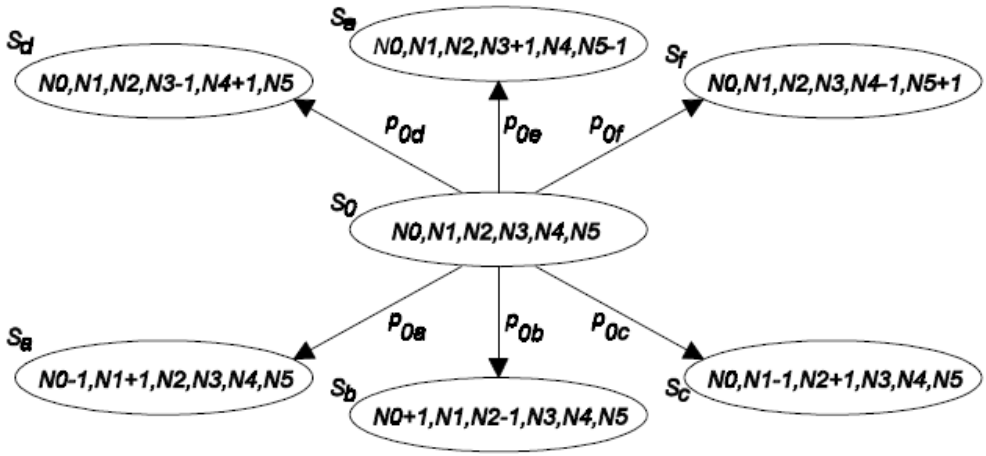
است. مثلا اگر $M = 3$ باشد، یک شبکه صف که یک سیستم ۴ – گره ای را مدل می کند، برای مدل کردن یک شبکه بزرگتر نیز کافی است (با سه ریسمان در هر پردازنده). با این حال، بخاطر تعامل بین کنترلر پردازنده – کش و کنترلر دایرکتوری، یک شبکه صف بسته مثل مدل ۱ قابل استفاده نیست. مدل های ساده تری از این نوع با یک زنجیره مارکف حل شده اند[۱۴]. تعداد زیاد حالت ها در مدل کنونی، استفاده کامل از راهکار زنجیره مارکف را غیر عملی می سازد. ما یک مدل تقریبی ارائه می دهیم که در آن یک گره خاص انتخاب می شود و بقیه گره های موجود و شبکه در دو صف خلاصه می شوند. نتایج این مدل، تطابق خوبی را با نتایج شبیه سازی از خود نشان می دهد. شبکه صف در شکل ۶ نشان داده شده است. گره ۰، گره انتخاب شده است که مثل مدل قبل، شامل یک پردازنده با کش، یک دایرکتوری و یک سرور کانال می باشد. دو زنجیره از پیغام ها، از این شبکه صف عبور می کنند. پیغام های متعلق به گره ۰، زنجیره C_0 و بقیه پیغام های متعلق به دیگر گره ها، زنجیره C_a را می سازند. یک پیغام متعلق به زنجیره C_0 از سرور پردازنده و سرور کانال می گذرد. بعد از خروج از سرور کانال، برای مدت زمانی مساوی تأخیر آن پیغام، وارد قسمت دیگر شبکه می شود. یک پیغام متعلق به C_a از یک گره در قسمت دیگر شبکه شروع به حرکت می کند و از کنترلر دایرکتوری و سرور کانال می گذرد. تأخیر این پیغام، مساوی زمانی است که در گره ۰ می گذراند. علاوه بر این، زمانی که یک پیغام از زنجیره C_a در قسمت دیگر شبکه صرف می کند، برابر زمانی است که یک پیغام از زنجیره C_0 در گره ۰ می گذراند(زمان تفکر[۵۵]). فرض می کنیم که این شبکه صف با یک زنجیره مارکف تقریب زده می شود و تعامل بین پردازنده و سرورهای دایرکتوری را با تنظیم احتمالات انتقال حالت زنجیره مارکف مدل می کنیم.

هر حالت، نشانگر تعداد پیغام های درون هر زنجیره در هر کدام از صف های شبکه می باشد و به صورت $(N_0, N_1, N_2, N_3, N_4, N_5)$ می باشد که N_0, N_1, N_2، به ترتیب تعداد پیغام های زنجیره C_0 در پردازنده، دایرکتوری و سرورهای کانال می باشند. بطور مشابه، N_3, N_4, N_5 تعداد پیغام های متناظر در زنجیره C_a می باشند. شکل ۷ انتقال های حالت از یک حالت S_0 را نشان می دهد (که تمام تعداد پیغام ها، غیر صفر است). انتقال از حالت S_0 به حالت های S_a, S_d، آنهایی هستند که تعامل بین پردازنده و دایرکتوری وجود دارد. رفتن از سرور پردازنده در گره ۰ با نرخ p_{0a} انجام می شود که برابر $1/R$ است اگر هیچ

مشتری C_a در سرور دایرکتوری نباشد و برابر α_0/R است اگر حداقل، یک مشتری C_a در سرور دایرکتوری موجود باشد.



شکل ۶: مدل ۲، شبکه صف



شکل ۷: مدل ۲، انتقال حالت

$$p_{0a}=\begin{cases}\frac{1}{R} & \text{if } N_3=0\\ \frac{\alpha_0}{R} & \text{if } N_3>0\end{cases} \quad (3)$$

بطور مشابه، مشتری های C_a در سرور دایرکتوری، با نرخ p_{0d}، آنرا ترک می کنند که برابر با $1/L$ یا α_1/L است که L توسط معادله ۱ داده شده است و $\alpha_0 + \alpha_1 = 1$ است.

$$p_{0d}=\begin{cases}\frac{1}{L} & \text{if } N_0=0\\ \frac{\alpha_1}{L} & \text{if } N_0>0\end{cases} \quad (4)$$

پیغام ها، با نرخ های متفاوتی بسته به اینکه به کدام زنجیره تعلق دارند، سرور کانال را ترک می کنند[۵۶]. پیغام های زنجیره C_0 با نرخ p_{0c} ترک می کنند که برابر $0.25/T'$ است و پیغام های زنجیره C_a با نرخ p_{0f} ترک می کنند که برابر $0.75/T'$ است زیرا به اندازه سه برابر پیغام های بیشتری در زنجیره C_a نسبت به زنجیره C_0 وجود دارد. T' زمان سرویسی است که در معادله ۲ نشان داده شده است. نهایتا، انتقال از حالت S_0 به حالت های S_b, S_e نشانگر ورود یک پیغام زنجیره C_0 به صف پردازنده و ورود یک پیغام زنجیره C_a به صف دایرکتوری می باشد که به ترتیب وقتی زمان تأخیر یا تفکر سپری می گردد، رخ می دهند. نرخ های ترک متناظر، به تعداد پیغام های موجود در قسمت باقیمانده شبکه بستگی دارد. مخصوصا اگر N_2 پیغام زنجیره C_0 بیرون از گره ۰ باشند، آنها در یک سرور دایرکتوری دور (یا کانال متناظر) قرار دارند. تمام N_2 پیغام می توانند در دایرکتوری های دور مشابه یا متفاوت باشند. هر پیغام بطور متوسط، در Q_L، زمان LAT را صرف می کند که تأخیر متوسطی است که پیغام با آن مواجه می باشد. با درنظر گرفتن احتمالات اینکه N_2 پیغام در Q_L با یک، دو یا سه دایرکتوری سرویس می شوند، داریم:

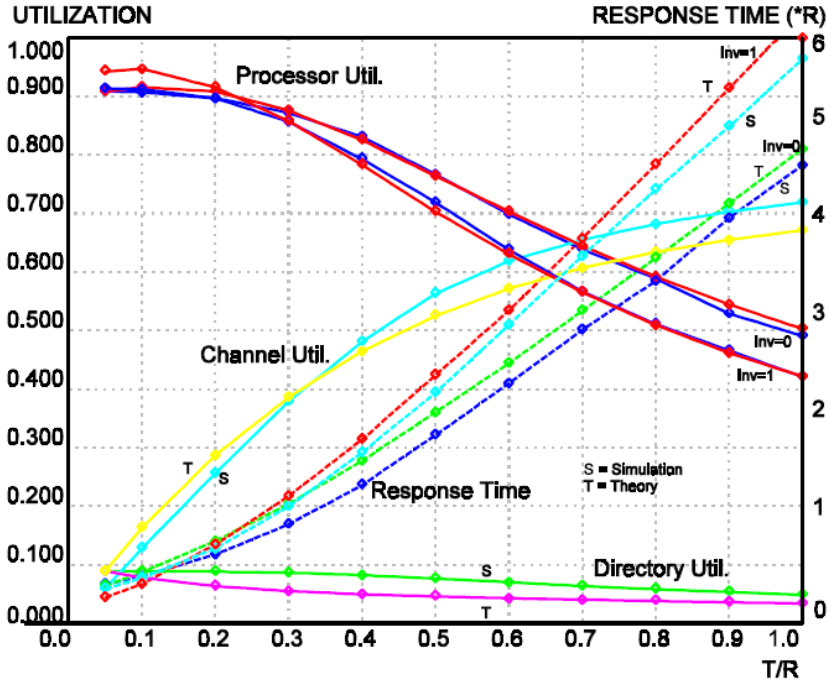
$$p_{0b}=\begin{cases}\frac{1}{LAT} & if\ N_2=1\\ \frac{1}{LAT}\frac{1}{3}+\frac{1}{LAT}\frac{2}{3} & if\ N_2=2\\ \frac{1}{LAT}\frac{3}{27}+\frac{1}{LAT}\frac{18}{27}+\frac{1}{LAT}\frac{6}{27} & if\ N_2=3\end{cases} \quad (5)$$

بطور مشابه، هر پیغام در Q_T، زمان متوسط TNK را صرف می کند که زمان متوسط تفکر است. با احتساب احتمالات اینکه N_5 پیغام در Q_T در یک، دو یا سه پردازنده قرار می گیرند داریم:

$$p_{0e}=\begin{cases}\frac{1}{TNK} & if\ N_5=1\\ \frac{1}{TNK}\frac{1}{3}+\frac{1}{TNK}\frac{2}{3} & if\ N_5=2\\ \frac{1}{TNK}\frac{3}{27}+\frac{1}{TNK}\frac{18}{27}+\frac{1}{TNK}\frac{6}{27} & if\ N_5=3\\ \frac{1}{TNK}\frac{1}{3^{N_5-1}}+\frac{1}{TNK}\frac{2^{N_5}-2}{3^{N_5-1}}+\frac{1}{TNK}\left(1-\frac{1}{3^{N_5-1}}-\frac{2^{N_5}-2}{3^{N_5-1}}\right) & if\ N_5>3\end{cases} \quad (6)$$

با این فرض که یک زنجیره مارکف با احتمالات انتقال، آنگونه که در بالا آمده است، معماری SOME – Bus را توصیف می نماید، می توانیم احتمالات حالتها را با تکنیک های استاندارد محاسبه نماییم. دو پارامتر در حل زنجیره مارکف وجود دارد: تأخیر(LAT) پیغام های زنجیره C_0 وقتی بیرون گره ۰ هستند و زمان تفکر(TNK) پیغام های زنجیره C_a وقتی بیرون از گره ۰ می باشند. ازآنجا که مقادیر این پارامترها از روی احتمالات حالت زنجیره مارکف محاسبه می گردد، ما از یک روش تکراری استفاده می نماییم: ابتدائا تأخیر LAT، برابر زمان متوسط انتقال کانال، 'T، و زمان تفکر، TNK، برابر زمان متوسط اجرای ریسمان قرار داده می شود؛ سپس احتمالات حالت محاسبه می گردند و مقادیر جدید LAT و TNK محاسبه گشته و این فرایند تکرار می شود تا زمانیکه همگرایی مشاهده شود. این فرایند همگرا می شود چون اگر مقداری برای یک پارامتر از مقدار صحیح بیشتر باشد، حل احتمالات زنجیره مارکف، به مقداری منجر می شود که که از مقدار انتخاب شده کمتر است (و بطور مشابه برای مقادیر انتخاب شده ای که از مقادیر صحیح کمتر هستند). مثلا اگر مقدار انتخاب شده برای پارامتر LAT از مقدار صحیح بیشتر باشد، پیغام های بیشتری تمایل دارند در Q_L بمانند که به صف های کوچکتری در دایرکتوری و کانال گره ۰ منجر می شود و

مقدار جدید LAT کوچکتر خواهد بود. در آزمایشات، در ۸ – ۶ مرحله، همگرایی مشاده شده است.



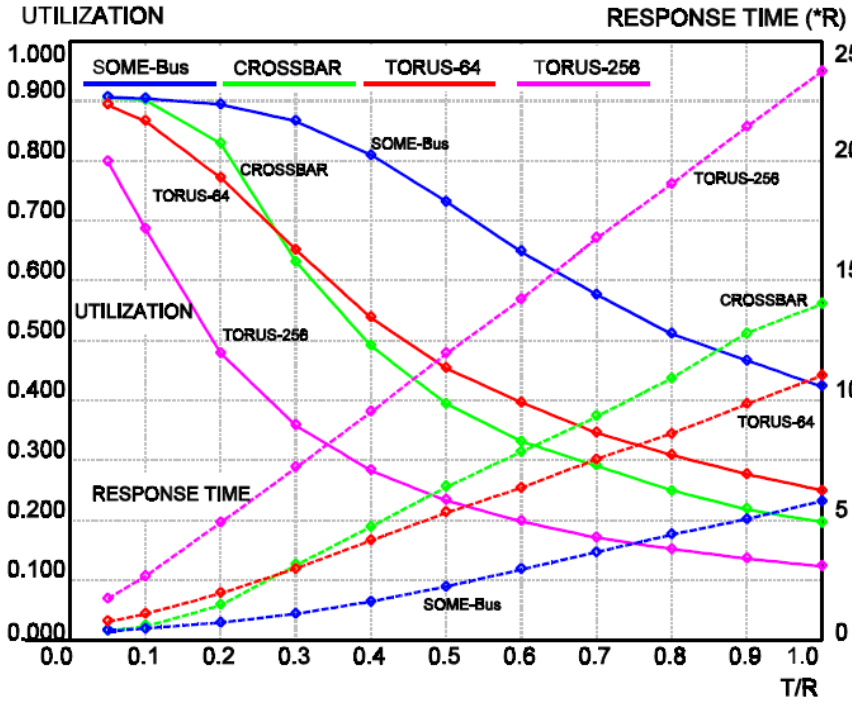
شکل ۸: مدل ۲، کارایی SOME – Bus

با این فرض که تمام زمان های پردازش و انتقال، توزیع هندسی دارند، شکل ۸ کارایی معماری SOME – Bus را بر حسب بازدهی پردازنده، بازدهی دایرکتوری، بازدهی کانال و زمان پاسخ نشان می دهد. زمان پاسخ، بازه زمانی بین موقعی است که یک عدم برخورد کش باعث می گردد تا یک پیغام در صف کانال خروجی قرار بگیرد تا موقعی که داده یا وصول متناظر وارد صف ورودی گردد. وقتی هر دو فعال باشند، به کنترلرهای کش و دایرکتوری، نصف پهنای باند حافظه می رسد($\alpha_0 = \alpha_1 = 0.5$). همانند قبل، نسبت زمان متوسط انتقال به زمان متوسط اجرای ریسمان بین ۰،۰۵ و ۱ تغییر می کند که برای بررسی رفتار سیستم تحت تنظیمات و نرخ های عادی عدم برخورد کش، کافی می باشد. نتایج کارایی از روی مدل نظری صف و شبیه سازی یک معماری SOME – Bus با ۶۴ گره بدست آمده است. شکل، کارایی SOME – Bus را برای دو حالت نشان می دهد: الف) وقتی همه پیغام های فرستاده شده به گره دور، پیغام های داده بخاطر عدم برخورد خواندن سراسری است و ب) وقتی ترکیبی از پیغام های درخواست – داده و درخواست – مالکیت وجود دارد.

در حالت دوم، احتمال اینکه یک پیغام، یک پیغام داده(بخاطر یک عدم برخورد خواندن سراسری) باشد، $p_{rd} = 0.8$ است؛ احتمال اینکه یک درخواست – مالکیت باشد(بخاطر یک عدم برخورد نوشتن سراسری) برابر 0.2 است. احتمال اینکه یک بلوک در حالت مشترک یافت شود برابر $p_{sh} = 0.8$ است. وقتی کنترلر دایرکتوری، یک درخواست – مالکیت به یک بلوک در حالت مشترک دریافت می کند، پیغام های نامعتبرسازی را به گره هایی که یک کپی از آن بلوک دارند، ارسال می دارد. در شکل ۸ تعداد این نامعتبرسازی ها، برابر ۱ فرض شده است. در حالت (الف)، وقتی همه پیغام ها، پیغام های درخواست – داده به بلوک های مشترک هستند، هیچ نامعتبرسازی ارسال نمی گردد. این دو حالت در شکل ۸ با Inv = 0, Inv = 1 نشان داده شده اند. در هر دو حالت، مخصوصا در بازدهی پردازنده، تطابق خوبی بین نتایج نظری و شبیه سازی وجود دارد. هر چقدر نرخ عدم برخورد افزایش می یابد، نسبت T/R نیز افزایش می یابد. پیغام ها زمان بیشتری (در مقایسه با زمان متوسط اجرای ریسمان، R) در شبکه صرف می کنند که به افزایش زمان پاسخ و بازدهی کانال منجر می گردد. در حضور پیغام های درخواست – مالکیت، زمان سرویس دایرکتوری بخاطر فرستادن نامعتبرسازی ها و جمع آوری پیغام های وصول نامعتبرسازی، افزایش می یابد. افزایش زمان پاسخ باعث کاهش بازدهی پردازنده می شود.

نتایج شبیه سازی و مقایسه

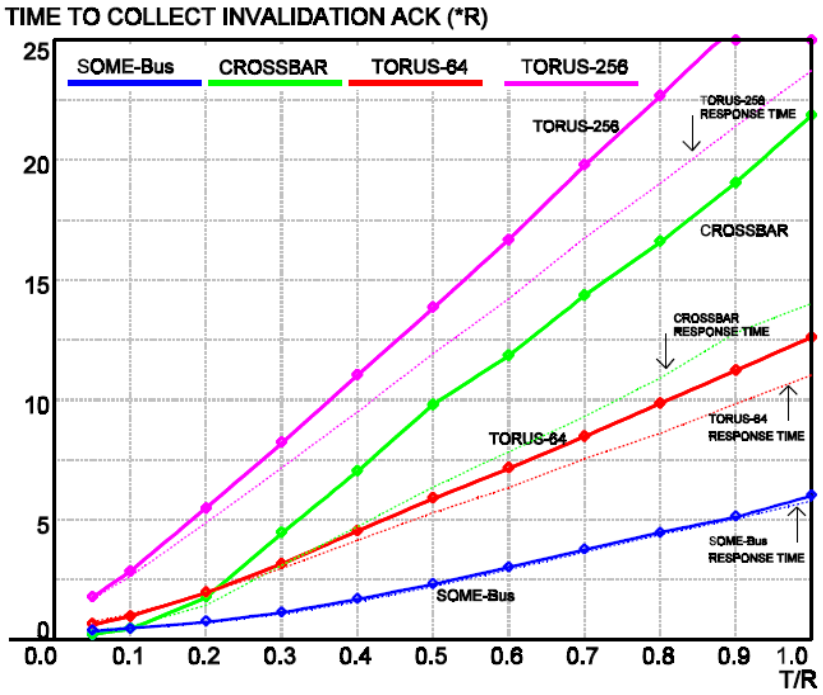
شکل ۹ نتایج شبیه سازی (بازدهی پردازنده و زمان پاسخ) را برای مقایسه کارایی معماری SOME – Bus با کارایی توروس دو – بعدی و circuit – switched crossbar و تحت مفروضات مدل ۲ نشان می دهد. این مفروضات، مانند مفروضات مدل ۱ هستند و علاوه بر آن فرض می کنیم که عملکرد دایرکتوری در هر سه معماری، با کش تداخل[۵۷] دارد.



شکل ۹: مدل ۲، بازدهی پردازنده و زمان پاسخ

در هر سه معماری، پردازنده هر گره، یک برنامه با M = 3 ریسمان اجرا می کند. یک درخواست – مالکیت به یک بلوک مشترک به سه نامعتبرسازی منجر می شود. نتایج برای سیستم هایی با ۶۴ و ۲۵۶ گره مقایسه شده اند. مانند آزمایشات قبلی، زمان اجرای ریسمان دارای توزیع هندسی با میانگین ۱۰۰ واحد زمانی می باشد. اندازه پیغام طوری انتخاب شده است که زمان انتقال T در محدوده ۵ تا ۱۰۰ واحد زمانی ثابت باشد(بجای اینکه توزیع هندسی داشته باشد) و نسبت T/R در بازه ۰,۰۵ تا ۱ ثابت بماند. بخاطر کاهش واریانس زمان انتقال پیغام، بازدهی پردازنده در مقایسه با بازدهی نشان داده شده در شکل ۸، اندکی افزایش می یابد. معماری های SOME – Bus و crossbar، مقیاس پذیری کامل را از خود نشان می دهند و همچنین هیچ تفاوتی در بازدهی پردازنده بین سیستم های ۶۴ – گره ای و ۲۵۶ – گره ای وجود ندارد. در مقابل، توروس ۲۵۶ – گره ای بازدهی پردازنده بسیار کمی حتی در مقادیر کوچک T/R از خود نشان می دهد. با اینکه توروس چهاربرابر کانال بیشتری نسبت به SOME – Bus دارد، بخاطر این واقعیت که wormhole ها حتی در ترافیک متوسط، متمایل به بلوکه شدن می باشند، عمدتا، تأخیر بسیار زیاد می گردد.

شکل ۱۰ زمان مورد نیاز برای جمع آوری پیغام های وصول نامعتبرسازی و زمان پاسخ کلی (خط چین) برای هر معماری را نشان می دهد. در SOME – Bus پیغام های وصول نامعتبرسازی، نسبت به باقی پیغام ها، تأثیر بیشتری در تأخیر ندارند. در مقابل، این پیغام ها، در معماری های دیگر، نقش بسیار مهمی در ایجاد تأخیر دارند با اینکه فقط قسمت کمی (۲۰%) از کل پیغام ها، درخواست های مالکیت است.



شکل ۱۰: زمان جمع آوری پیغام های وصول نامعتبرسازی

مراجع

[1] Nemawarkar, S. S., Govindarajan, R., Gao, G.R., Agarwal, V.K., "Analysis of multithreaded multiprocessors with distributed shared memory", IEEE Symp. Parallel Distributed Processing, 1993, pp. 114-121.

[2] Mabbs, S.A., Forward, K.E, "Performance Analysis of MR-1, a Clustered Shared-Memory Multiprocessor", Journal of Parallel and Distributed Computing, Feb. 1, 1994, v 20, n 2., p. 158.

[3] Sterling, T. Merkey, P., Savarese, D., "Improving Application Performance on the HP/Convex Exemplar", Computer, Dec. 01 1996, v 29, n 12, p. 50.

[4] Grujic, A., Tomasevic, M., Milutinovic, V., "A Simulation Study of Hardware- Oriented DSM Approaches", IEEE Parallel & Distributed Technology, Spring 1996, v 4, n 1, p. 74.

[5] Agarwala, A., and Das, C. R. "Experimenting with a Shared Virtual Memory Environment for Hypercubes", Journal of Parallel and Distributed Computing, Sep. 1, 1995, v 29, n 2, p. 228.

[6] Bhuyan, L. N., Iyer, R. R., Kumar, M., "Performance of Multistage Bus Networks for a Distributed Shared Memory Multiprocessor", IEEE Transactions on Parallel and Distributed Systems, Jan. 1 1997, v 8, n 1, p. 82.

[7] Dahlgren, F. Stenstrom, P, "Evaluation of Hardware-Based Stride and Sequential Prefetching in Shared-Memory Multiprocessors", IEEE Transactions on Parallel and Distributed Systems, Apr. 1 1996 v 7 n 4, p. 385.

[8] Bhuyan, L., "Generalized Hypercube and Hyperbus Structures for a Computer Network", IEEE Trans. Computers, v. C-33, n. 4, pp. 323-333, 1984.

[9] Ould-Khaoua, M., "Comparative Evaluation of Hypermesh and Multi-Stage Interconnection Network", Computer J., 1996 v. 39 n. 3 p. 232.

[10] Szymanski, T., "Hypermeshes: Optical Interconnection Network for Parallel Computing", J. Parallel and Distributed Computing, Apr. 1, 1995 v. 26 n. 1 p. 1.

[11] Lindquist R.G., J. H. Kulick, W. E. Cohen, R. K. Gaede, E. Wells, M. Abushagur, D. Shen, C. Katsinis, S. T. Kowel, "An Optoelectronic Design of the Simultaneous Optical Multiprocessor Exchange Bus (Some-Bus)", SPIE Proceedings v. 3025, San Jose, Feb. 12-14, 1997.

[12] Willick, D. L., Eager, D. L., "An Analytic model of Multistage interconnection networks", ACM SIGMETRICS, May 1990, pp. 192-202.

[13] Adve, V., "Performance Analysis of Mesh Interconnected Networks With Deterministic Routing", IEEE Transactions on Parallel and Distributed Systems, Mar. 1, 1994, v 5, n 3, p. 225.

[14] Katsinis, C., "Distributed-shared-memory support on the Simultaneous Optical Multiprocessor Exchange Bus", 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'98), Montreal, Canada , July 1998.

پیوست ۱- واژگان

1. Distributed – Shared – Memory
2. Contention
3. Management Agents
4. Coherent
5. Remote
6. Consistency
7. Performance
8. Reordering
9. Pipelining
10. Overlapping
11. Synchronize
12. Utilization
13. Global
14. Load Balancing
15. Cache Miss
16. Multistage Bus
17. Prefetch
18. Hyper – Mesh
19. Outstanding
20. Discrete – Time Markov Chain
21. Self – Loop
22. Single – Step State Transition
23. Singular
24. Arbitration Delay
25. Simultaneous Optical Multiprocessor Exchange Bus
26. Bottleneck
27. Header
28. Barrier Processing
29. Length Monitoring
30. All – to – All
31. Hypercube
32. Torus
33. Patterning
34. Yield
35. Snooping
36. Update
37. Invalidation
38. Multithreaded
39. Thread
40. Host
41. Owner
42. Acknowledgement
43. Read Miss
44. Write Miss
45. Data – Request
46. Ownership – Request
47. Shared
48. Modified
49. Multi – Chain Closed Queuing Network

50. Data – Acknowledge
51. Ownership – Acknowledge
52. Write – Back
53. Iterative
54. Direction
55. Think Time
56. Departure
57. Interference