بسم الله الرحمن الرحيم

دانشگاه اراک

دانشکده ی فنی و مهندسی

گروه کامپیوتر

پایان نامه ی کارشناسی رشته ی کامپیوترگرایش نرم افزار

طراحی مفهومی و مدل واقعی سیستم های سخت افزاری پردازش موازی

استاد راهنما:

دکتر سعید مقیمی

پژوهشگر:

نسرین قاسمی

آبان ماه ۱۳۸۸

کلیه ی حقوق مادی مرتبت بر نتایج مطالعه ها،ابتکارها

و نوآوری های ناشی از تحقیق موضوع این پایان نامه

متعلق به دانشگاه اراک است

دانشگاه اراک

دانشکده ی فنی و مهندسی

گروه کامپیوتر

پایان نامه ی کارشناسی رشته ی کامپیوترگرایش نرم افزار خانم نسرین قاسمی

طراحی مفهومی و مدل واقعی سیستم های سخت افزاری پردازش موازی

در تاریخ توسط هیات داوران زیر بررسی و با درجه ی به تصویب نهایی رسید.

۱-استاد راهنمای پایان نامه دکتر سعید مقیمی با درجه ی علمی امضا

۲-استاد داور داخل گروه مهندس بهمن هاشمی با درجه ی علمی امضا

امضای مدیرگروه

چکیده:

با توجه به افزایش قدرت و کاهش روز افزون قیمت کامپیوتر های شخصی و همچنین رشد و توسعه ی شبکه های کامپیوتری، در سال های اخیر تلاش گسترده ای برای ساخت سیستم های پردازش موازی مبتنی بر کامپیوتر های شخصی صورت گرفته است. این سیستم ها به دلیل استفاده از سخت افزارهای آماده و ارزان دارای هزینه کارایی بهتری نسبت به ابرکامپیوترها هستند و در ضمن نرم افزارها و سیستم عامل های این پردازنده ها، فراوان ودر دسترس می باشند. نکته ی اصلی در این سیستم ها، نحوه ی اتصال این کامپیوترها به همدیگر و نحوه ی ارتباط آنها می باشد. مشخصات عمومی که مورد نظر همه سیستمهای موازی می شود: کارایی بالا وهزینه کم،کارایی قابل اطمینان و طراحی قابل گسترش است.در راستای این اهداف دو روش برای ساختن سیستم های موازی بوجود آمد:چند پردازنده ای و چند کامپیوتری. در سیستمهای چند پردازنده ای، پردازنده ها از طریق یک حافظه مشترک که قابل دسترسی توسط همه پردازنده ها می باشد، با هم ارتباط بر قرار می کنند. حال آنکه درسیستمهای چند کامپیوتری، هر پردازنده دارای یک حافظه اصلی است و پردازنده فقط قادر به دسترسی به حافظه خودش است. ارتباط در این سیستمها، معمولا بصورت مبادله پیام انجام می گیرد.

کلید واژه ها: لینوکس، پردازش موازی، کلاسترینگ،گره های کلاستر،گره های محاسبه ای.

فهرست مطالب

عنوان	صفحه
فصل اول- مقدمه ای بر سیستم های پردازش موازی	۱
۱.۱. تعریف پردازش موازی	۲
۲.۱. مشخصات پردازش موازی	۴
۳.۱. تکنیک ها و روش های پردازش موازی	۴
۱.۳.۱. چند پردازنده ای متقارن(SMP)	۴
۲.۳.۱. دسترسی حافظه ی غیر یکنواخت (NUMA)	۶
۳.۳.۱. پردازنده های موازی کلان(MPP)	۷
۴.۳.۱. سیستم های توزیع شده	۸
۵.۳.۱.کلاسترها (Clusters)	۸
فصل دوم – کلاسترینگ	۹
۱.۲. تاریخچه ی مختصری از کلاسترینگ	۹
۲.۲. انواع کلاسترها	۱۱
۱.۲.۲. تعادل بار (Load Balancing)	۱۱
۲.۲.۲. قابلیت بالا (High Avalibility)	۱۳
۳.۲.۲. محاسبات شبکه ای (Grid Computing)	۱۷
۴.۲.۲.کلاستر هایی برای محاسبات (Clusters For Computing)	۱۷

۳.۲. تفاوت کلاستر های محاسباتی و محاسبات شبکه ای باکلاستر HA و تعادل بار ۱۸

فصل سوم – معماری کلاسترلینوکس High-Performance ۱۹

۱.۳. سخت افزار گره ۱۹

۲.۳. سیستم عامل گره ۲۰

۳.۳. نرم افزار کلاسترینگ ۲۰

۴.۳. اتصالات داخلی ۲۲

فصل چهارم– ساخت یک کلاستر محاسبه ای موازی نمونه ۲۹

۱.۴. مشخصات سیستم ۳۰

۱.۱.۴.پیکربندی سخت افزار ۳۰

۲.۱.۴. پیکربندی نرم افزار ۳۲

۲.۴. سیستم عامل لینوکس ۳۳

۱.۲.۴. هسته ی لینوکس چیست؟ ۳۴

۲.۲.۴. ساخت تصویر هسته ۳۴

۳.۲.۴. ساخت سیستم فایل ریشه ۳۵

۴.۲.۴. روال راه اندازی لینوکس ۳۸

۱.۴.۲.۴. راه ا ندازی هسته ی سرور ۳۸

۲.۴.۲.۴. راه ا ندازی هسته ی کلاینت ۳۸

۳.۴. جزئیات روال ساخت ۳۸

فصل پنجم- کلاسترهای نمونه ۴۳

۱.۵. کلاستر ALICE ٤٤

مراجع

فهرست شکل ها

فصل اول- مقدمه ای بر سیستم های پردازش موازی

شکل۱-۱: پردازش متوالی از یک وظیفه ی منفرد ۲

شکل ۱-۲: پردازش موازی : اجرا و انجام کارهای جزئی به طور موازی ۲

شکل۱-۳ : پردازش متوالی کارهای مستقل چندتایی ۳

شکل۱-۴ : پردازش موازی: اجرا و انجام کارهای مستقل به طور مشابه و مساوی ۳

شکل ۱-۵: تکنولوژی چند پردازنده ای متقارن ۵

شکل ۱-۶ :تکنولوژی پردازش های موازی کلان ۸

فصل دوم – کلاسترینگ

شکل۲-۱: تاریخچه ای از supercomputing ۱۰

شکل ۲-۲: بخش هایی از کلاستر load balancing ۱۲

شکل۲-۳ : بخش هایی از کلاستر high availability ۱۴

فصل سوم – معماری کلاسترلینوکس High-Performance

شکل۳-۱: مسیر یک بسته بین دو گره محاسباتی ۲۳

فصل چهارم- ساخت یک کلاستر نمونه

شکل۴-۱ :پیکربندی سخت افزار کلاستر ۳۰

فصل پنجم- کلاسترهای نمونه

شکل۴-۱: معماری کلاستر ALICE ٤٣

فهرست جداول

جدول ۴-۱:مشخصات نود اصلی کلاستر ۳۲

جدول ۴-۲:مشخصات نود سرور سیستم فایل NFS و ROOT– NFS ۳۲

جدول ۴-۳:مشخصات نود کلاینت ۳۳

جدول۴-۴:مشخصات هاب ۳۳

فصل اول– مقدمه ای بر سیستم های پردازش موازی

۱.۱. تعریف پردازش موازی:

پردازش موازی[۱]، یک کار بزرگ را به چند کار کوچکتر تقسیم می کند و کار های کوچکتر را به طور همزمان در چند گره اجرا می کند. در نتیجه، کار بزرگتر سریعتر کامل می شود.

توجه : یک گره، یک پردازشگر جداگانه و اغلب در یک دستگاه جداگانه است. به هر حال، پردازشگر-های چند جانبه می توانند در اختیار یک دستگاه منفرد باشند.

برخی کارها می توانند به طور موثر تقسیم شوند وکاندیداهای خوبی برای پردازش موازی هستند. برای مثال، در یک بانک با یک تحویل دار، مشتریان باید برای راه انداختن کار خود، به صف شوند. با دو تحویل دار، کار انتظار مشتریان می تواند به طور موثر بین دو تحویل دار تقسیم شود بنابراین مشتریان، دوبرابر سریعتر خدمات رسانی می شوند. این یک نمونه ای است که پردازش موازی، یک راه حل موثر و کارآمدی است. همه ی کارها نمی توانند به طور موثر و کارآمد تقسیم شوند. فرض کنید که برای مثال قبلی رئیس بانک، باید همه ی درخواست های وام را تایید کند. در این مورد پردازش موازی ضرورتاً خدمات رسانی را افزایش نمی دهد.

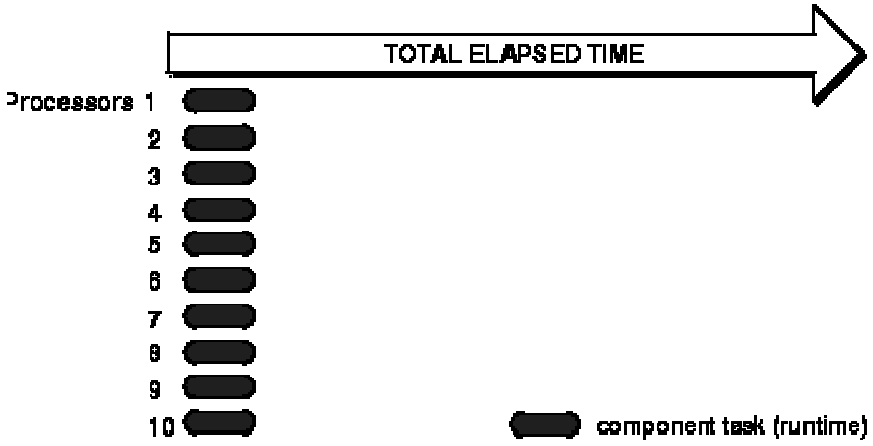
---

[۱] Parallel processing

مهم نیست که چند تحویل دار برای پردازش و رسیدگی وام ها موجود هستند همه ی درخواست ها باید برای تایید رئیس بانک در نوبت قرار بگیرند. هیچ پردازش موازی نمی تواند بر این مانع ذاتی غلبه کند. تصاویر زیر پردازش متوالی یک سوال مشابه را با پردازش موازی همان سوال مورد مقایسه قرار می دهد. شکل۱-۱ پردازش متوالی را نشان می دهد که در آن یک پرسش به عنوان یک کار منفرد اجرا می شود.

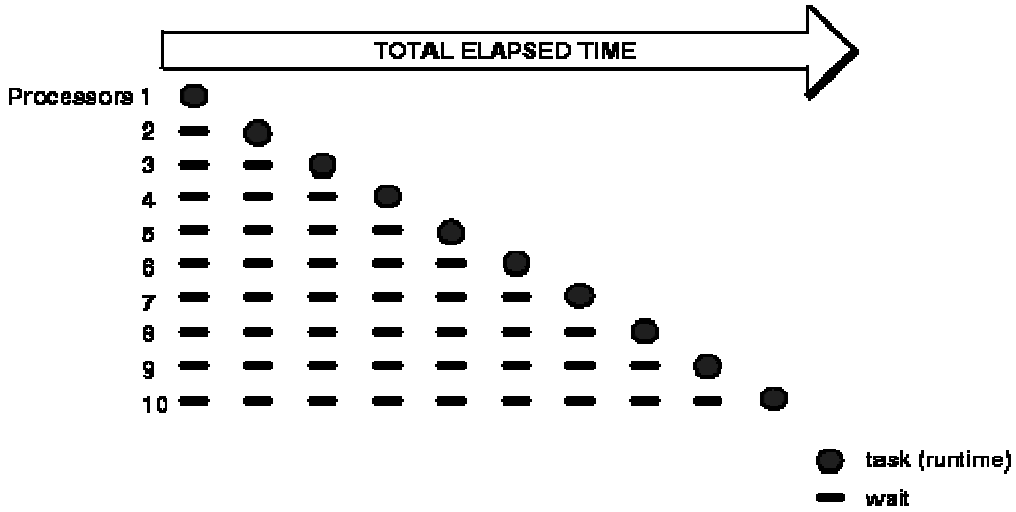
شکل۱-۱: پردازش متوالی از یک وظیفه ی منفرد

شکل ۱-۲ پردازش موازی را نشان می دهد که در آن یک سوال به چند کار کوچکتر تقسیم می شود و هر کار جزئی در یک نمونه ی جداگانه اجرا می شود .



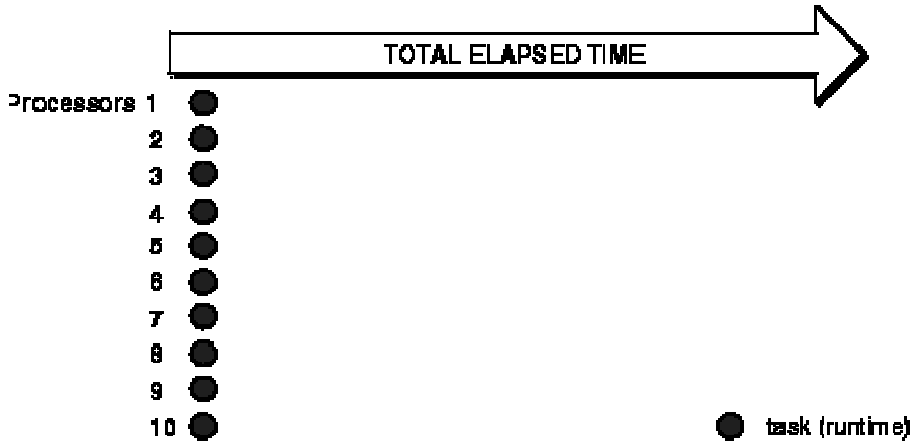
شکل ۱-۲ : پردازش موازی : اجرا و انجام کارهای جزئی به طور موازی

این تصاویر پردازش متوالی با پردازش موازی کارهای مستقل چند تایی را از محیط های کامپیوتری پردازش درون خطی (OLTP) را مورد مقایسه قرار می دهد. شکل۱-۳ پردازش متوالی کارهای مستقل چندتایی را نشان می دهد.



شکل ۱-۳ : پردازش متوالی کارهای مستقل چندتایی

شکل ۱-۴ پردازش موازی کارهای مستقل و جداگانه را نشان می دهد .



شکل۱-۴ : پردازش موازی: اجرا و انجام کارهای مستقل به طور موازی

در پردازش متوالی، کارهای مستقل، بر سر یک منبع منفرد، رقابت می کنند. فقط کار ۱، بدون انتظار کشیدن راه اندازی می شود. کار ۲ باید منتظر بماند تا کار ۱کامل شود. کار۳ باید منتظر بماند تا کارهای ۱و ۲ تکمیل شود، و به همان ترتیب. اگرچه تصویر، کارهای جداگانه را به اندازه ای مشابه و یکسان نشان می دهد ولی مقدار و اندازه ی کارها ممکن است فرق داشته باشند. بر عکس، اگر شما یک سرور موازی در یک چند

پردازشگر متقارن داشته باشید می توانید CPU قویتری را برای کارها اختصاص دهید، بسته به اینکه CPUs شما، چگونه تقسیم می شوند. هر کار جداگانه، بلافاصله در پردازشگر خاص خود اجرا می شود، و زمان ا نتظاری را در بر نمی گیرد.

برای استفاده از پردازش موازی می توان دلایل زیر را بر شمرد. از دلایل اولیه ی پردازش موازی صرفه جویی در زمان، توانایی حل مساله ی بزرگتر در یک بازه ی زمانی ثابت و انجام چند کار به طور هم زمان می باشد. از دیگر مزایای استفاده از پردازش موازی بهره وری از منابع غیر محلی می باشد. از نگاهی دیگر با استفاده از منابع محاسباتی ارزا نتر و موجود به صورت موازی می توان در پرداخت هزینه برای خرید ابر کامپیوتر ها صرفه جویی کرد. وقتی از یک کامپیوتر به صورت منفرد استفاده می شود محدودیتی در حداکثر مقدار حافظه ی آن وجود دارد که برای حل مسائل بزرگتر استفاده از حافظه ی چندین کامپیوتر می تواند راهی برای غلبه بر این محدودیت باشد.

۲.۱. مشخصات یک سیستم موازی:

یک سیستم پردازش موازی مشخصات زیر را دارد :

- هر پردازشگر در یک سیستم می تواند به طور همزمان کارها را انجام دهد.
- ممکن است کارها همزمان سازی شوند.
- گره ها معمولاً منابعی مانند داده ها، دیسک ها و دیگر دستگاه ها را تقسیم می کنند.

۳.۱. تکنیک ها یا روش های پردازش موازی:

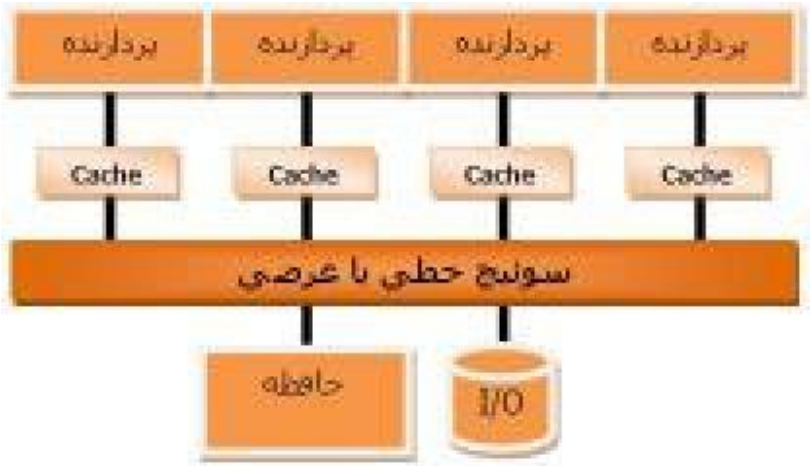
۱.۳.۱. چند پردازنده ای متقارن[۲]

امروزه، سیستم های چند پردازشگر، عموماً از نوع متقارن هستند. این به آن معنا است که یک پردازشگر،

[۲] Symmetric multi processor

هیچ اهمیت بیشتری نسبت به بقیه ندارد، و همه ی منابع برای همه ی پردازشگر ها به طور یکسان در دسترس هستند. سیستم هایی از این نوع، چند پردازشگر متقارن یا SMP نامیده می شوند. یک کامپیوتر منفرد، چندین CPU دارد ولی دارای فضای حافظه ی تقسیم شده ی منفرد و امکانات I/O تقسیم شده است.

ایده ی پشت SMP، به نحوی آشکار عبارت است از تقسیم کردن مساله ی کامپیوتر به فرایند های همزمان و اجازه دادن به آنها برای اجرا در پردازشگر های جداگانه در همان دستگاه. در یک دستگاه SMP، خود سیستم عامل، مسئول تقسیم کردن فرایند های فردی ایجاد شده با یک تقاضا در میان CPU های موجود است. دستگاه های SMP در برنامه ها و سیستم عامل هایی بهتر استفاده می شوند که از فرایند کم حجم یا باریک استفاده می کنند. ویندوزهای NT بسیار محدود و خلاصه است و فرایند های لینوکس نسبتاً کم حجم هستند. به طوری که هر دو مقیاس و اندازه نسبتاً خوبی در سخت افزار SMP هستند. سیستم های SMP با دو یا چند پردازشگر، برای ساخت، نسبتاً ساده هستند. هر چیزی ماورای آن نسبتاً سخت می شود، زیرا همه ی پردازشگر ها نیاز دارند تا بتوانند به همه ی منابع حافظه و I/O دسترسی پیدا کنند. در بیش از چهار پردازشگر این منابع تقسیم شده شروع به در مضیقه قرار گرفتن می کنند و افزودن CPU های بیشتر کاهش دادن ارسال ها را ایجاد می کند. شکل ۱-۵ معماری SMP را نشان می دهد که از منابع سیستم، مانند حافظه وI/O زیرسیستم که همه پردازنده ها می توانند بصورت مساوی به آنها دسترسی پیدا کنند، بصورت اشتراکی استفاده می کند[5].



شکل ۱-۵: سیستم چند پردازنده ای متقارن

۲.۳.۱ . دسترسی غیر یکنواخت به حافظه[۳]

کامپیوتر ها یSMP از یک طرح تقسیم حافظه استفاده می کند در این که هر پردازشگر، سطح یکسانی از دسترسی به همه ی حافظه ی فیزیکی درکامپیوتر را دارد. چنین طرحی به عنوان دسترسی حافظه ی یکنواخت یا UMA شناخته می شود. NUMA (دسترسی غیر یکنواخت به حافظه) تکنیک یا روش پیچیده تری است که به چندین پردازشگر در یک کامپیوتر چند پردازشگر امکان می دهد، حافظه ی محلی را به روشی موثرتر نسبت به SMP ساده تقسیم کند. هر CPU، دستیابی سریع و مستقیمی به منطقه ی حافظه منفرد دارد ولی می تواند به دیگر مناطق حافظه، در سیستمی با دسترسی فوری کمتری دست پیدا کند.

ایده ی اصلی NUMA دادن یک امتیاز به پردازشگر های خاص در دستیابی به محدوده ی حافظه ی فیزیکی ارائه شده است. شما می توانید دستگاه NUMA را به عنوان نوعی مرحله ی میانی بین دستگاه های SMP ساده و در مقیاس وسیعی سیستم های موازی تصور کنید. دستیابی به هر بخش از حافظه در یک سیستم NUMA امکان پذیر است و آن دقیقاً ممکن است برای دستیابی به بخشی از آدرس های حافظه نسبت به دیگران زمان بیشتری را بگیرد. به هر حال زمان دستیابی به حافظه ی غیر محلی هنوز هم نسبت به دستیابی به دیسک یا شبکه ی I/O سریع تر خواهد بود .

سیستم در یک دستگاه NUMA کاملاً پیچیده است.آن غالباً به عنوان یک شبکه با اتصالات زیاد به گذرگاه اجرا می شود. شاید شما اصطلاح ccNUMA را ببینید،که نشان می دهد که سیستم، اتصال نهانگاه را حفظ می کند. وقتی یک CPU در حال دسترسی به حافظه است، نهانگاه ورودی به همه ی پردازشگر های دیگر باید کنترل شود برای اطمینان دادن به اینکه آنها داده هایی را که در حال بازیابی هستند تغییر نداده اند.

سیستم های NUMA سعی می کنند مساله ی اصلی را با محاسبه ی موازی بهینه سازی کنند: یعنی ارتباطات داخلی پردازشگرها. درکلاسترها و سیستم های موازی کلان، هزینه های جاری ارتباط دادن بین پردازشگر ها، تا حدودی بالاست، زیرا ارتباط باید در سرتاسر یک شبکه ی از چند نوع، حرکت کند.

---

[۳] Non-Uniform Memory Access

NUMA از یک گذرگاه حافظه ی پر سرعت برای ارتباط از طریق حافظه ی تقسیم شده استفاده می کند. در حالی که سرعت دستیابی به حافظه ی غیر محلی، به زیادی سرعت دستیابی به حافظه ی محلی نیست، ولی آن بسیار بیشتر از ارتباط در سرتاسر شبکه است. دستگا ه های NUMA خیلی خوب به اوج تعداد زیادی از پردازشگرها می رسند بنابراین آنها می توانند گاهی اوقات رقیب عملکرد سیستم های موازی کلان برای محاسبه ی ظرفیت باشند. جنبه ی منفی این است که همانطورکه شما ممکن است تصور کنید طراحی این دستگاه ها الگوریتم ها ی پیچیده ای را بر اساس طرح ها ی میانجی و زمانبندی های بسیار دقیق نانو شامل می شوند. بنابراین آنها گرایش دارند که دستگاه ها یا ماشین های نسبتاً گرانی باشند[5].

۳.۳.۱. پردازنده های موازی کلان[۴]

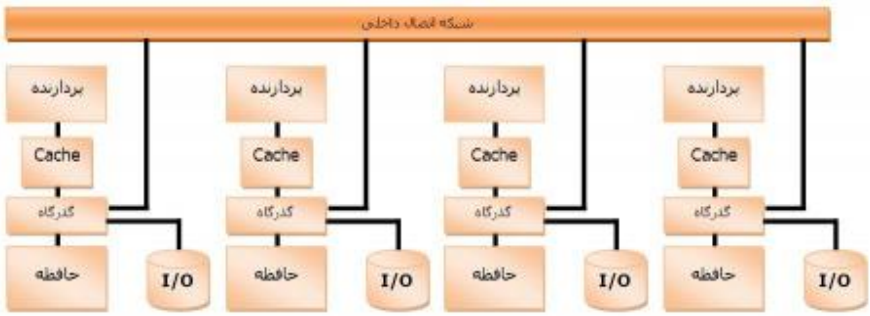
پردازش موازی کلان پراهمیت ترین دنیای محاسبه ی موازی است. در مدلMPP هر گره شامل یک پردازشگر جداگانه با منابع اختصاصی خاص خود می شود. ایده ی یک سیستم MPP تقسیم کردن یک مساله ی کامپیوتری به قسمت هایی است که می تواند به طور جداگانه و کم و بیش به طور مستقل از یکدیگر محاسبه شوند. از طرفی دیگر، طراحی سیستم واحد هایی دارد که نسبتاً مستقل هستند. سیستم های موازی کلان معمولاً برای عملیات ها ی وسیع کامپیوتری و بسیار نهایی استفاده می شوند.

MPP ارتباط بسیار نزدیکی با کلاسترینگ دارد، ولی هر گره در یک سیستم MPP معمولاً قابلیت های کامل I/O را ندارند. بنابراین هر گره در یک سیستم MPP ممکن است یک کامپیوتر کارامد مستقل نباشد. یک سیستم MPP معمولاً بزرگتر از یک کلاستر نمونه است. یکی از مشکلات در MPP این است که برنامه ها باید برای سیستم های موازی به طور خاصی نوشته شوند. دو رابط برنامه نویسی کاربردی[۵] عمومی هستند که استفاده می شوند: MPI , PVM. این ها بر روی تقسیم کردن یک مساله به چند بخش عمده تاکید می کنند که می تواند به طور موازی محاسبه شوند. بنابراین اگر مساله ی مورد حل نمی تواند به این روش تقسیم شود یک سیستم MPP کمک زیادی نخواهد بود [5].

---

[۴] Massively Parallel Processors

[۵] Application Programming Interface



شکل ۱-۶ :تکنولوژی پردازنده های موازی کلان

۴.۳.۱. پردازش توزیع شده [۶]

پردازش توزیع شده احتمالاً کمترین مشخصه از همه ی اصطلاحاتی است که ما در اینجا پوشش داده ایم پردازش توزیع شده به این معناست که بخش هایی از کار در حال انجام در جاهای مختلفی انجام می شوند رایج ترین نمونه از پردازش تقسیم شده سرور/ مشتری است. سرور یک کار خاصی برای انجام دارد درحالی که مشتری بخش دیگری از کار را انجام می دهدکه عموماً کار نشان دادن اطلاعات به کاربراست. یک سیستم تقسیم شده نسبت به کلاستر به طور ضعیف تری متصل می شود. در واقع دیدن هر جفت یا اتصالی درکل دشوار است.

۵.۳.۱.کلاستر ها

یک کلاستر، گروهی ازسیستم های کامپیوتری شخصی است که می تواند برای ظاهر شدن به عنوان یک سیستم کامپیوتری، ایجاد شود. در حالی که این تعریف می تواند ساده به نظر برسد، ولی چندین تکنولوژی مشابه دیگر وجود دارد. تفاوت های بین تکنولوژی ها می تواند تا حدی نامحسوس باشد[5] .

---

[۶] Distributed processing

فصل دوم- کلاسترینگ

۱.۲. تاریخچه مختصری از کلاسترینگ

کلاسترینگ کامپیوتر از دهه ی۱۹۸۰در انواع مختلفی دورگرفته است، که منشا در پایگاه دیجیتالی VAX دارد. سیستم عامل VMS و سخت افزار VAX برای فراهم کردن سرویس های کلاستری ترکیب شدند. این کلاستر های VAX قادر بودند، منابع سخت افزاری، مانند فضای دیسک را تقسیم کنند، و قادر بودند منابع کامپیوتری را برای کاربران چندگانه فراهم کنند.

کلاسترینگ تقریباً هم دوره با محاسبات کامپیوتر بزرگ[۷] است. از زمان های گذشته، توسعه دهنده ها می خواستند برنامه های کاربردی بسازند که قدرت محاسبه ی بیشتری را نسبت به قدرتی که یک سیستم منفرد می توانست تولید کند نیاز داشت. بنابراین برنامه های کاربردی،که می توانست از ویژگی محاسبه ی موازی برای اجرا شدن روی پردازشگر های چندگانه استفاده نماید، ایجاد شد. کلاستر ها هم چنین می توانند قابلیت اطمینان یک سیستم را افزایش دهند، تا کنون از بین رفتن هیچ یک از بخش ها باعث غیر قابل استفاده شدن تمام سیسستم نشده است.

بعد از کامپیوتر های بزرگ، مینی کامپیوتر ها و ایستگاه های کاری اجرایی توسط فروشندگانی مانند Hewlett- Packard، Tandem و Silicon Graphics به کلاستر ها پیوسته بودند. این سیستم ها سخت افزار اختصاصی و سخت افزار اتصالات و پروتکل های ارتباطات اختصاصی را استفاده می کردند.

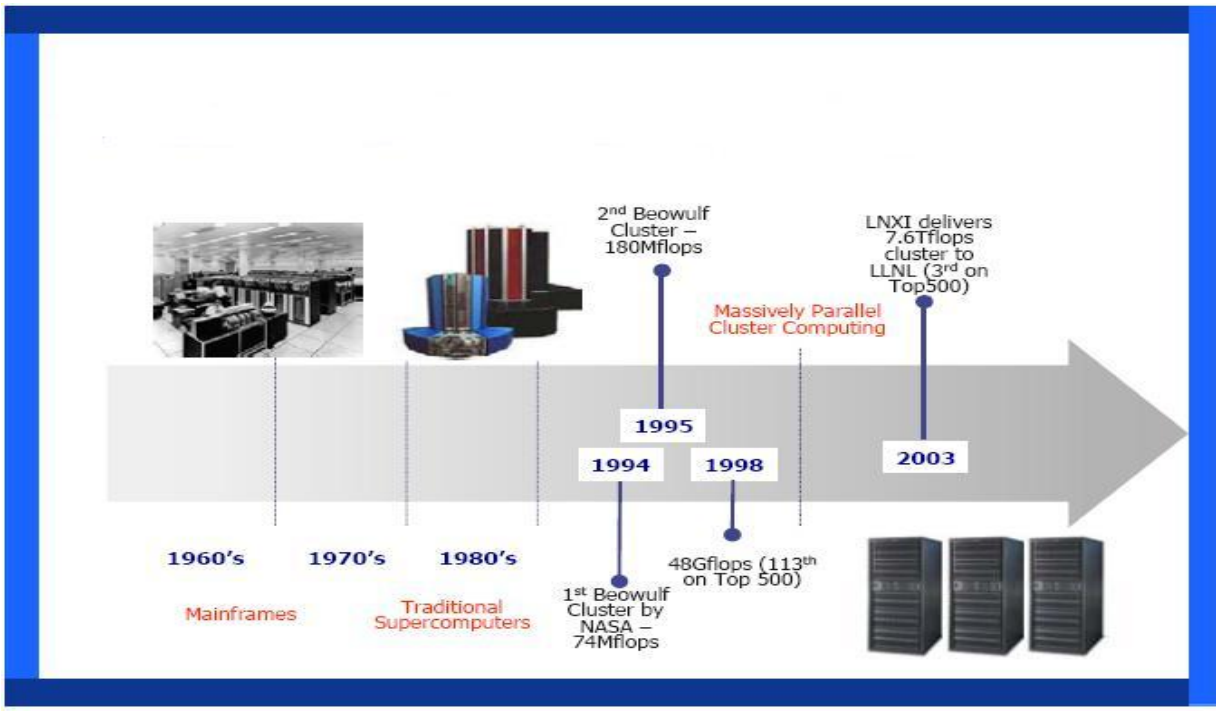
چالش در برابر کلاستر های اختصاصی به این دلیل است است که سخت افزار و نرم افزار آنها تمایل به پرهزینه بودن دارند، واگر فروشنده از پشتیبانی محصول باز ایستد، کاربران ممکن است با مشکل مواجه شوند.

---

[۷] Mainframe

میکروسافت و ناول، هر دو کلاستر های ساخته شده با ویندوز های NT و سیستم عامل های NetWard و سخت افزار مربوط به خودشان را پیشنهاد می کنند. اگرچه هیچ یک از این محیط های کلاسرینگ پیشنهاد شده گسترش نیافتند، ایده ی استفاده از سخت افزار Off-The Shelf کلاستر هایی می سازد که مطابق با این روش بودند.

هم اکنون تعداد بسیاز زیادی از بزرگترین کلاستر ها مبتنی بر سخت افزار PC های استاندارد که اغلب لینوکس را اجرا می کنند وجود دارند. یکی از راه حل های کلاسترینگ مشهور مبتنی بر لینوکس Beowulf است که توسط Donald Becker و Thomas Sterling در سال۱۹۹۵برای NASA ساخته شد. اولین کلاستر Beowulf شامل مجموعه ای از ۱۶۴۸۶ PC بود که با اترنت[۸] به هم متصل شده بودند امروزه سازمانها و شرکت های بسیاری وجود دارند که کلاستر هایی از نوع Beowulf را ارائه می دهند[4] .



شکل۲-۱: تاریخچه ای از supercomputing

---

[۸] Ethernet

۲.۲. انواع کلاسترینگ:

چهار نوع دسته بندی وجود دارد که در آن هر نوع یک نیاز خاصی را برآورده می کند. آنها عبارتند از : تعادل بار[۹] ، قابلیت بالا[۱۰]، محاسبات شبکه ا ی[۱۱] و کلاسترهایی برای محاسبات[۱۲].

۱.۲.۲. تعادل بار

این نوع کلاستر، کامپیوترها (معمولاً به عنوان گره ها اشاره می شوند) را وصل می کند که در آن هر کامپیوتری، بار کاری منحصر به فرد را فراهم می کند. هرگره یک ظرفیت کاری خاصی خواهد داشت به طوری که هر کامپیوتر می تواند به طور کارامد کار کند و گره در عملکرد یا کار اضافی، کار نخواهد کرد. مزیت تعادل بار این حقیقت است که آن همیشه برای کار کردن، بهینه می شود در حالی که نقص و عیب آن نیاز بالاتر حفظ و نگهداری است. اگر یک گره کار نخواهد کرد، کلاسترینگ امکان پذیر نخواهد بود. یک نوع کلاسترینگ ایده آل برای ا طمینان از پایداری و ثبات، تعادل بار یا ظرفیت است.کلاسترینگ از طریق تعادل بار یا ظرفیت اساساً شکلی از ارتباط بین کامپیوترهاست که در آن بار یا ظرفیت اضافی ناچاراً تقسیم میشود.اگرچه یک کامپیوتر منفرد می توانست پردازش های یکسانی را فراهم کند، ولی وضعیت سخت افزار یک کامپیوتر منفرد یا یک گره هرگز برای رسیدگی کردن به درخواست های داده های قابل توجه و پردازش، کافی نخواهد بود. در پاسخ و واکنش به درخواست های ثابت، کامپیوتر های متعددی برای مهار کردن قدرت هر پردازش به هم وصل می شوند. با تقسیم بار کاری، زمان پردازش افزایش می یابد و در خواست های داده های قابل توجه و مراحل می توانند امکان پذیر شوند. تعادل بار یا ظرفیت نوعی از کلاسترینگ و اغلب متفاوت با کلاسترینگ قابلیت بالا(HA) است.

دسته بندی HA نوعی ارتباط بین گره ها است که در آن یک گره ی اولیه پشتیبانی می شود با تعدادی

---

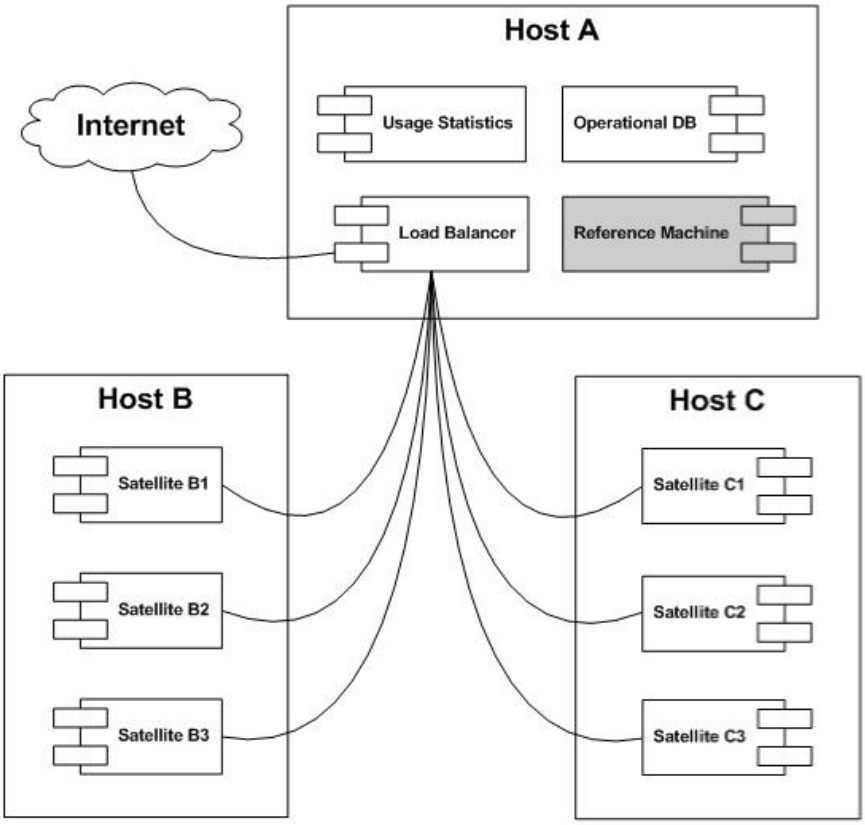
[۹] Load balancing

[۱۰] High availability

[۱۱] Grid computing

[۱۲] Clusters for computing

از گره ها در موردی که گره ی اولیه نقص پیدا کند. هدف HA و تعادل بار برای یک چیز است - ثبات و پایداری. آرایش و ترکیب بندی آنها اطمینان می دهد که ویژگی ها ی یک برنامه ی کاربردی همیشه موجود خواهد بود یا داده ها همیشه به هنگامی که مشتری به آنها نیاز دارد، آنجا خواهد بود. هدف HA برای انجام آن با داشتن پشتیبان های چندتایی است در حالی که تعادل ظرفیت یا بار، از ثبا ت و پایداری اطمینان می دهد ولی گره های چندتایی را برای کار در همان زمان به دست می آورد. شکل ۲-۲ کلاستر تعادل بار و تقسیم بار را نشان می دهد.



شکل ۲-۲: کلاستر load balancing

۱.۱.۲.۲ .انتقال ظرفیت یا بار کاری

یک ویژگی که برجسته بودن تعادل بار را ایجاد می کند قابلیت آن برای رسیدگی به نقص هاست. هر دستگاهی همیشه به میزان تلفات خود خواهد رسید و تنها روش برای رسیدگی به آنها داشتن یک راه حل

در حالتی است که این امر اتفاق می افتد. در تعادل بار یا ظرفیت بار کاری گره ی معیوب فوراً در دیگر گره ها دوباره توزیع و تقسیم می شود. با توزیع بار کاری قابلیت و راندمان ادامه می یابد زیرا گره ها فقط تغییرات نسبتاً کوچکی را احساس خواهند کرد. انتقال بار کاری زمانی انجام می شود که گره ای نقص پیدا کند همان زمان که تعادل کننده ی بار به گره هایی وصل می شود که شبیه یک نبض کار می کند که عملکرد هر گره را نشان می دهد. در یک فاصله ی از پیش تعیین شده تعادل کننده ی بار هر گره را کنترل خواهد کرد. و اجرا کننده ی شبکه در حالتی که یک گره نقص پیدا کند هشیار خواهد بود. تعادل کننده ها در کلاستر تعادل بار توانایی و قابلیت انتقال بار کاری به گره های دیگر را برای اجتناب بیشتر از تاخیر پردازش و عملیات دارند[6].

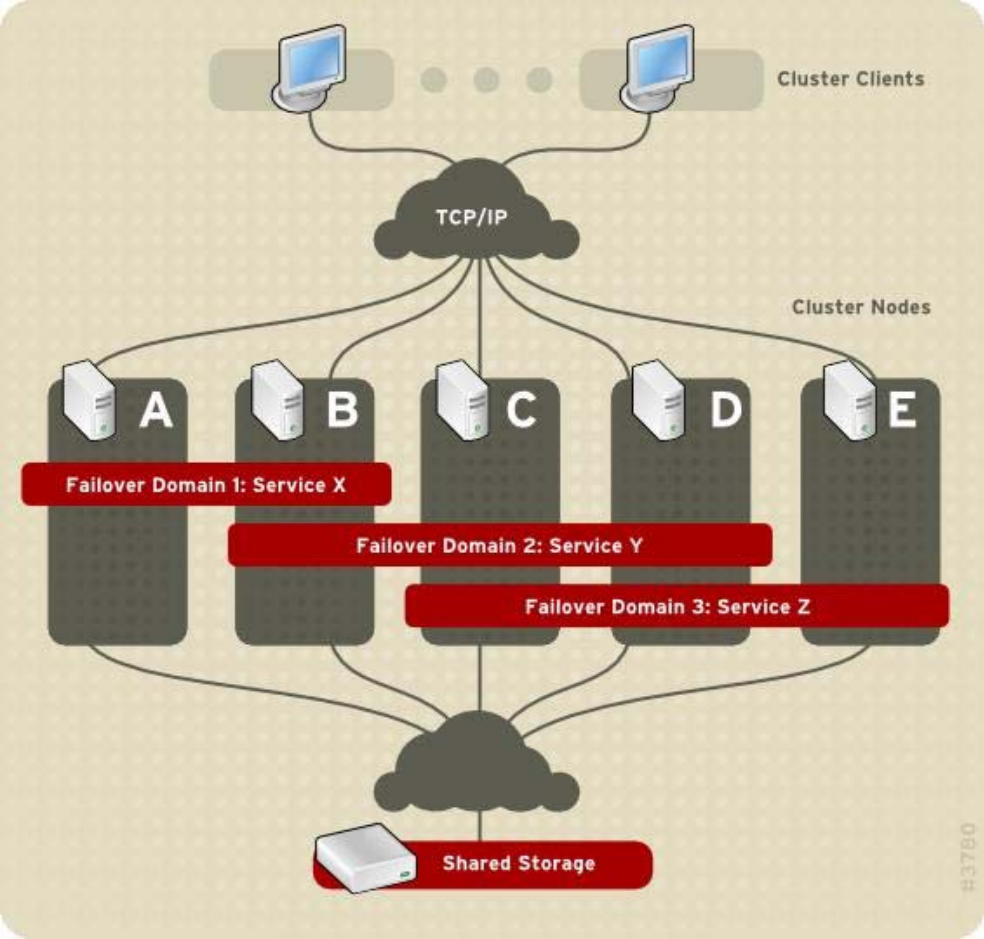
۲.۱.۲.۲. نیاز های سخت افزاری:

برای تعادل بار جهت کار سخت افزار اضافی مورد نیاز است که در آن هدف منحصر به فرد آن عبارت است از نمایش دائمی گره ها و تقسیم دوباره ی کاری در صورتی که یکی از گره ها دچار نقص شود. برخی این امر را به عنوان یک نقص و عیب برای کلاستر تعادل بار تصور می کنند زیرا آن به هزینه ی اضافی احتیاج خواهد داشت دقیقاً برای اطمینان از این که گره ها به خوبی کار خواهند کرد. ولی این سخت افزار اضافی واقعاً یک ترکیب بندی بهتری به ویژه برای تاجرانی است که ترافیک در مقیاس وسیعی را برای داده ها و خدمات کامپیوتری خود انتظار دارند. با کمک یک سخت افزار اضافی گره ها می توانند کاملاً بر اطمینان از این تاکید کنند که عملکرد ها به خوبی کار می کنند یا داده ها به محض امکان انتقال خواهند یافت. از طرفی کلاستر HA برای نمایش اولیه و مسلط شدن بر گره ی اولیه نیاز به گره هایی خواهند داشت آن هم در صورتی که گره ی اولیه دچار نقص شود. تعادل بار یا ظرفیت شاید در مقایسه با کلاستر HA هزینه ی بیشتری داشته باشد ولی سخت افزار اضافی، گره ها را در این طرح کلاستری بهینه سازی خواهد کرد[6] .

۲.۲.۲. قابلیت بالا

قابلیت بالا بر توانایی گره هایی تاکید می کند که کاملاً تحت هر شرایطی موجود می باشند. عملکرد و کارها می تواند توسط هر گره ای انجام و اجرا شود که در آن هر گره ای یک کمک و تقویت برای گره ی دیگر می شود در حالتی که چیزی اشتبا ه رخ می دهد. قابلیت و کارایی بالا هم چنین به عنوان کلاستر های معیوب یا کلاستر های با سطح افزونگی بالا اشاره می شود. ا لبته مزیت آن عبارت است از وعده ی ثبات و پایداری آن.

به هرحال این نوع کلاستر ممکن است پتانسیل کامل گره ها را به هنگامی که برخی می توانند تنها به عنوان کمک به کار گرفته شوند، استفاده نکند[6].در شکل ۲-۳ بخش هایی از کلاستر قابلیت بالا نشان داده شده است.



شکل ۲-۳: بخش هایی از کلاستر high availibility

۱.۲.۲.۲. انواع کلاستر(HA)

کلاستر HA می تواند بر اساس عمل و کار گره ها متفاوت باشد. عمومی ترین شکل و نوع کلاستر HA، آرایش و ترکیب بندی فعال/ فعال است. در این نوع کلاستر HA، گره ها برای تقسیم بار کاری برنامه ریزی می شوند. به هر حال وقتی یک گره برای کار دچار نقص می شود، بار کاری گره ی معیوب در دیگر گره ها توزیع می شود تا این که گره ی معیوب، ثابت شود.

حالت بعدی آرایش و ترکیب بندی فعال/غیر فعال است.گره ها در این آرایش همگی عملکردهای اضافی بر سازمان دارند که، آنها در حالتی که هر مشکلی بر می خیزد، مسئولیت گره ی اولیه را بر عهده خواهند گرفت. اساساً، وقتی گره ها دیگر فقط، کمک و حمایت کننده هستند تنها چندین گره در حال کار و عمل خواهند بود.

N+1 به عنوان آرایش و ترکیب بندی فعال/غیر فعال- با یک برگشتن در نظر گرفته می شود. گره های فعال زمانی بار اضافی یکسان را تقسیم می کنند که یک گره ای که قابلیت انجام هر چیزی را دارد، وجود دارد. به هر حال، این نوع گروه، به طور فعال استفاده نمی شود و تنها برای جایگزینی گره ای استفاده می شود که در کار کردن نقص دارد.

N+M می تواند به عنوان پرهزینه ترین نوعN+1، در نظر گرفته شود. به جای گره ی دیگری برای جایگزینی گره ی معیوب، یک سرور کمکی یا ذخیره مورد استفاده قرار می گیرد. M به سرور های بیشتری اشاره می کند که به تعادل گره کمک خواهند کرد.

وقتی که کلاستر HA، N را برای آرایش گره ی ۱ استفاده می کند،گره ی اضافی برای قابل قبول شدن از لحاظ جهانی، ولی تنها در حد خاصی، آرایش داده می شود.N+1 جایگزین گره ی معیوب می شود در حالی که N تا ۱ تنها یک آرایش و ترکیب موقتی است.

آرایش و ترکیب بندی N تا N تقریباً شبیه آرایش و ترکیب بندی فعال/ فعال است. به هر حال یک گره ی اضافی برای هر گره ی معیوب ایجاد می شود تا بدون رفتن به حالت ذخیره به آرایش و ترکیب بندی اصلی و اولیه باز گردد.

۲.۲.۲.۲.نیاز های آرایش و تر کیب بندی :

اگرچه انواع مختلفی از آرایش ها و ترکیب بندی ها ی HA وجود دارد، ولی احتیاجات کلی درHA برای اطمینان از موفقیت آن وجود دارد.

- سهولت عملیات شروع/ توقف :

این کار می تواند دستی یا از طریق یک برنامه باشد ولی آنچه مهم است این است که باید عملیاتی وجود داشته باشد که بلافاصله در صورت مشکل قابل توجه، یک گره را متوقف کند.

- قابلیت و کارایی شبکه ی منطقه ی ذخیره سازی

این برای اطمینان از قابلیت داده ها برای همه ی گره هاست.

- توانایی برای حالت ذخیره

آرایش و ترکیب بندی HA، باید قابلیت و توانایی ذخیره ی دائمی وضعیت کلاستر را داشته باشد به طوری که اخیرترین وضعیت به هنگامی که گره کار می کند، بتواند اجرا و انجام شود. این هم چنین اطمینان می دهد که شبکه داده ها را حتی اگر آن دچار نقص شود، حفظ خواهد کرد[6].

۳.۲.۲.۲.بیش از فقط داده/عمل

کلاستر قابلیت بالا، فقط در مورد داده ها و کارهای تقسیم شده نیست. اگرچه کار و نقش اصلی کلاستر HA، اطمینان از این است که داده ها برای کاربران در دسترس است و وظیفه برای کار کردن ادامه می یابد، ولی آرایش و ترکیب بندی های سخت افزاری نیز باید تقسیم شوند. تقسیم قدرت و ارتباطات کابلی نیز باید اجرا شوند. بدون این اتصالات فیزیکی، کلاستر HA هرگز عمل نخواهد کرد یا برای اطمینان دادن از راندمان دائمی به هنگامی که بیشتر گره ها برای کار دچار نقص خواهد شد، شکست خواهد خورد [6] .

۳.۲.۲. محاسبات شبکه ا ی

این نوع کلاستر به عمل و رفتار گره ها اشاره می کند. در محاسبات شبکه ا ی،گره ها، به هنگامی که آنها فقط یک خروجی را فراهم می کنند، نسبتاً با یکدیگر ارتباط دارند.

۴.۲.۲.کلاستر هایی برای محاسبات:

گره ها در این نوع کلاستر، به طورگسترده ای با هم ارتباط دارند همان طورکه آنها برای محاسبات و برنامه های گسترده ای استفاده می شوند. چون یک گره تنها، قابلیت سخت افزاری فراهم کردن داده های مورد نیاز در سر وقت ندارد، تعدادی از گره ها به منظور ایجاد نتایجی که مورد انتظار و به موقع هستند به یکدیگر وصل می شوند.

۳.۲. تفاوت کلاستر هایی برای محاسبات و محاسبات شبکه ا ی با کلاستر HA و تعادل بار

کلاستر HA و تعادل بار دو نوع کلاستر هستند ولی اغلب برای اهداف تجاری استفاده می شوند. آنها یک تفاوت چشمگیر با کلاستر های محاسباتی و محاسبات شبکه ا ی دارند آن هم به دلایل زیر:

توجه به دوام و پایداری:

اگرچه کلاسترینگ تماماً در حد کارایی و راندمان است، ولی اهداف تعادل بار و کلاستر HA برای نوع مختلفی از راندمان و بازدهی است.در عوض عملکرد بهتر کلاسترها در محاسبات، کلاسترها برای اطمینان از دوام و ثبات عملکردها و کارهای ساده، مورد استفاده قرار می گیرند. کلاستر های محاسباتی و محاسبات شبکه ای، از طرفی دیگر بر روی عملکرد و نقش تاکید می کند. اگرچه آن هنوز هم نیاز دارد که بسیار تثبیت و پایدار شود، ولی همیشه اولویت بر روی بهبود عملکرد و کار کلاستر ها است.

نرم افزار خاص:

کلاستر های محاسباتی و محاسبات شبکه ای، عملیات و پردازش خود را بر پایه ی نرم افزاری با عملکرد ها و کار های خاص قرار می دهند. نرم افزار نصب شده در این نوع کلاستر ها، برای نشان دادن رفتار و عمل گره ها مورد استفاده قرار می گیرد. در مورد تعادل بار و HA، آرایش و ترکیب بندی می تواند از طریق اتصالات سخت-

افزاری آنها مشخص و تعیین شود. اگرچه نرم افزار هایی وجود دارند که بار و ظرفیت را کنترل می کنند، ولی نرم افزار های اضافی می تواند از طریق گره هایی راه اندازی شوند که توسط کاربران فراهم می شوند.

بر پایه ی کارامدی:

کلاسترهای محاسباتی و محاسبات شبکه ای، غالباً برای نقش منفرد یا مجموعه ای از نقش ها که برای کلاستر ها از پیش تعیین می شوند استفاده می شوند. تعادل بار و HA، از طرفی دیگر می تواند برای تثبیت و پایداری سیستم های مختلف ارائه شوند همان طور که عمل و رفتار آنها توسط سخت افزار دیکته می شود یک نرو افزار ی برای کلاستر تعادل بار و HA، اختصاص داده می شود.

فصل سوم- معماری کلاسترلینوکس High-Performance

قبل از اسمبل کردن کلاستر تعدادی از انتخاب ها برای ساخته شدن نیاز است. کدام پردازشگرها را شما استفاده خواهید کرد؟ کدام سیستم عامل؟ کدام روش اتصال بین گره ها؟ کدام محیط برنامه نویسی؟ هر تصمیمی که گرفته می شود روی انتخاب های دیگر تاثیر خواهد گذاشت و بعضی از آنها احتمالا به وسیله ی کاربرد کلاستر ها اعمال خواهد شد.

۱.۳. سخت افزار گره

گره های کلاستر کار واقعی از کلاستر را انجام می دهند. اولین بخش در معماری کلاستر، سخت افزار گره است. این احتمالاً یکی از پردازشگر های AMD یا Intel خواهد بود، اما انتخاب های بسیاری وجود دارد، و آنها به تقاضای شما و فاکتورهای دیگر بستگی خواهد داشت، مانند ناحیه ی فیزیکی که شما برای کلاستر در دسترس دارید.

پردازشگر های AMD Opteron با معماری مستقیم بعضی مزیت ها را در مقایسه با پردازشگر Intel Xeon ارائه می دهد. مورد اول، کنترل کننده حافظه ی مجتمع AMD64 پردازشگر است که بیشترین پهنای را تولید می نماید. این ویژگی در مدت زمان دسترسی به حافظه تاثیر می گذارد که می تواند برای پردازشگر AMD Opteron دو برابر سریعتر باشد، تا زمانی که هسته ی سیستم عامل این ویژگی را پشتیبانی می کند.

پردازشگر Intel Xeon ، یک معماری حافظه ی تقسیم شده دارد که در آن کنترل کننده ی حافظه روی یک سیلیکان جداگانه قرار دارد. در یک پیکربندی چند پردازنده ای، کنترل کننده ی حافظه بین پردازشگر ها تقسیم می شود. مزیت دیگر این است که، پردازشگر AMD معمولاً برتری عملکر/قیمت را دارد. نتیجه ی آن درکمترین هزینه ی دارندگی و مالکیت می باشد. این مزیت خیلی مهم به نظر می رسد وقتی شما در حال ساخت یک کلاستر بزرگ هستید.

معمولاً دو نوع از گره ها وجود دارد،گره های اصلی،گره های محاسباتی. گره های اصلی معمولاً فقط برای یک بار برای شبکه ی عمومی قابل دیدن هستند، و به اجرای امنیت اضافی و نرم افزار کنترل دسترسی، سرور هایDHCP (ابزارهایی برای دادن IP به کلاینت های یک شبکه) و DNS(سیستم نامگذاری حوزه) و نرم افزار های پشتیبانی دیگر احتیاج خواهند داشت. آنها اساساً دست کم دو رابط شبکه دارند، یکی برای شبکه های عمومی برای ارتباطات خارجی و دیگری برای شبکه ی خصوصی برای کلاستر ها. کاربران می توانند گره های اصلی را با هاردیسک های بزرگ، حافظه یبیشتر وCPUs سریعتر و رابط های شبکه افزایش دهند[4].

۲.۳. سیستم عامل گره

از زمانی که کلاسترها با سیستم عامل های مختلف می توانند ساخته شوند، لینوکس به دلیل هزینه کم، تنوع زیاد ابزارهای موجود و سودمند، و تعداد بالای پژوهش های حاضر که برای آماده سازی کلاستر برای کارکردن انجام گرفته است، یک انتخاب محبوب است.

حتی درون لینوکس، چندین انتخاب برجسته وجود دارد، اساساً Red Hat و Suse. انتخاب لینوکس به نرم افزار کلاستر یا چیزی که یک عضو IT ممکن است در آن وارد باشد یا یک ویژگی که ممکن است نیاز باشد ممکن است بستگی داشته باشد.

به علاوه برای خود سیستم عامل، یک تعداد زیاد از نرم افزار های کمکی مانند کامپایلر ها، کتابخانه ها، فایل های پیکربندی، سرور های DHCP و DNS و نرم افزار های SSH یا RSH و غیره نیاز خواهد بود. گسترش و نگهداری تصاویر نرم افزار ها برای هر نوع از گره در کلاستر مهم است. هر گره ی پیرو به صورت صحیح با کلید های SSH، تنظیمات DHCP پیکربندی شده است. اشکال زدایی یکی از سخت ترین مراحل آماده سازی تقاضای کلاستر شده برای کارکردن مناسب است [4] .

۳.۳. نرم افزار کلاسترینگ

توزیع های متنوعی از نرم افزار کلاسترینگ وجود دارد که برای لینوکس قابل استفاده است، محدوده ای ازکامپایلرها برای محیط های کلاسترینگ کامل که سیستم عامل و همه ی درایور های لازم را در بر می گیرد.Rocks، یک توزیع کد باز ایجاد شده به وسیله ی دانشگاه کالیفرنیا درSan Diego و سیستم عامل تجاری Scyld Beowulf دو نوع از بهترین توزیع های شناخته شده ی کامل هستند.

در نتیجه برای انتخاب های دیگر، نرم افزار کلاسترینگ انتخابی شما به نیاز های دیگر شما بستگی خواهد داشت.

پایه های معماری:
بسته های کلاستر
بسته های کلاستر ساختمان کلاستر را به راحتی ایجاد می کنند
بسته های کلاستر بخش های لازم برای مدیریت کلاستر را در بر می گیرند:
Message Passing Interface (MPI or PVM)
Management tools (LSF, LVS, GRID Engline)
Batch/queue software (LSF, PBS)
Easy to use interface for installation and configuration

MPICH، یک پیاده سازی سبک از MPI، استاندارد برای کتابخانه های گذرنده ی پیام، وبه طور آزاد قابل استفاده است. نسخه های مختلفی از MPICH ممکن است برای پشتیبانی کامپایلرهای مختلف نیاز باشد، و نسخه های مختلفی از کامپایلرها ممکن است برای سازش با محیط های کلاسترینگ دیگر نیاز باشد.

پایه های معماری:
میان افزار
OSCAR – (Open Source Cluster Application Resources) is a collection of best-known components for building, programming, and using clusters.
SCYLD – Fee based Beowulf cluster management software & tools, now part of Penguin Computing
ROCKS - Open Source Beowulf cluster management software & tools collaborative development headed by University of San Diego.

اگر فقط یک تقاضا وجود دارد که روی کلاستر اجرا می شود، ایجاد یک توزیع تنها ، ممکن است امکان پذیر باشد. به هر حال اگر مانند بیشتر کامپیوتر های بزرگ، تولید یک سرویس برای بخش های مختلف لازم است، یک رویکرد انعطاف پذیر برای ارائه و ترکیب بندی کلاستر نیاز است.

محصولات تجاری، مانند محیط کلاسترینگ Scyld Beowulf، یک سیستم کامل برای ایجاد کلاستر ها تولید می کنند که یک کتابخانه ی MPI، زمانبند، و سرویس های مانیتورینگ را شامل می شود. که می تواند در میان نصب گره ی اصلی و گره ی محاسباتی تغییر داده شود، و آن ابزار های برنامه نویسی را در بر می گیرد. امروزه، رابط گذرنده ی پیام (MPI) در بازار حکم فرمایی می نماید و استاندارد برای گذر پیام است. اگرچه MPI برای بیشتر تقاضاها ی موازی رایج است، توسعه دهنده ها با یک چالش رو به رو هستند، واقعاً هر نوع از نیازهای اتصال یک پیاده سازی مشخص از استاندارد MPI است[4].

۴.۳. اتصال ها[۱۳]

بافت شبکه ای که گره های محا سباتی را دریک کلاستر برای حمل کردن مخابره پیام بین گره ای مرتبط می سازد، به عنوان اتصال کلاستری یا به طور ساده اتصال می باشد. واژه شبکه ی ناحیه ای سیستم (SAN) نیز گاهی اوقات برای دلالت کردن به یک اتصال کلاستری مورد استفاده قرار می گیرد. اگرچه محدوده ی SAN ها چیزی وسیع تر از اتصال های کلاستری می باشد. دقت کنید که یک اتصال به شبکه ی ناحیه ی محلی (LAN) استفاده کننده وصل نمی شود. آن به طور کامل تحت حوزه ی اجرایی از مدیر کلاستر می باشد.

انتخاب اتصال مناسب برای یک کلاستر خیلی مهم و برای درست شدن خیلی دشوار می باشد. از طرف دیگر، با وابسته بودن روی اتصالی که شما انتخاب می کنید، ممکن است به یک درصد مشخص از هزینه ی کلاستر تبدیل شود. تحقیق و تمرین درزمینه ی تکنولوژی اتصال کلاستر با عملکرد بالا به طور قابل توجهی بر پایه ی تکنولوژیهای اتصال MPP وتکنولوژیهای عمومی LAN وابسته می باشد. قبل از اینکه ما درمورد تکنولوژیهای ویژه به طور جزئیات، شروع به صحبت کنیم، اجازه دهید ابتدا یک مقدمه ی خلاصه ای را برای مقایسه و تطبیق تکنولوژیهای متعدد، شروع کنیم.

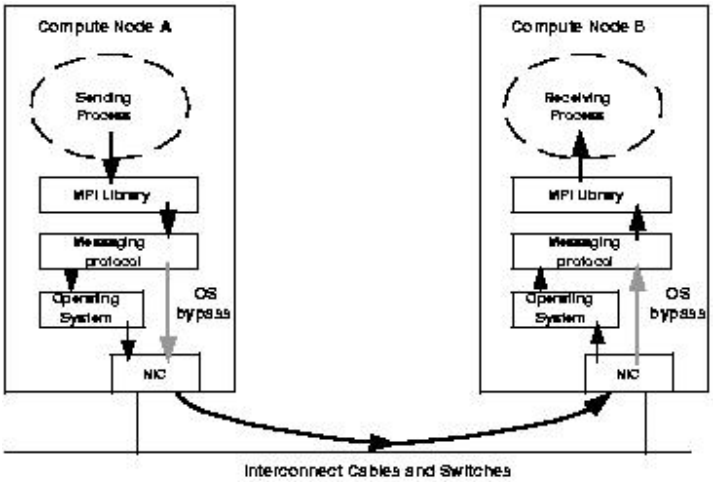
---

[۱۳] Interconnects

۱.۴.۳. پایه های اتصال (Interconnected Basice)

شکل ۳-۱ مسیر یک بسته را دریک کلاستر از یک گره محاسباتی به گره دیگر نشان می دهد. ما این مدل را به جهت توضیح دادن مفاهیم مرتبط دراین بخش ساده کرده ایم، برای مثال همه ی لایه های متعدد را در بین خود بافت اتصال نشان نداده ایم. یک فرایند، که قسمتی از یک تقاضا ی موازی می باشد درگره A اجرا می شود و در حال تلاش برای فرستادن یک پیام به فرایند دیگر می باشد که درگره B اجرا می شود.فرایند ارسال کردن پیام را بوسیله ی یک تماس به کتا بخانه ی برنامه نویسی برای مثال کتا بخانه MPI شروع می کند.

کتابخانه ی MPI پیام را به لایه ی پروتکل ویژه که برای اجرا کردن کتابخانه ی MPI مورد استفاده قرار می گیرد، می فرستد. سپس این لایه پروتکل پیام را یا به طور مستقیم و یا از طریق سیستم عملکرد ی روی شبکه ی فیزیکی عبور می دهد. درروی قسمت دریافت، حوادث درترکیب زمانی وارونه همانند آن در میزبان فرستادن اتفاق می افتد. سرانجام بسته را برای پرداز ش تقاضا فراهم می کند. دریک سطح فیزیکی، بسته از حافظه ارسالی کلاستر به کارت رابط شبکه (NIC) عبور می کند، سپس به کابل شبکه عبور می کند واز طریق سویچ ها انجام می شود، وبه NIC از کلاستر دریافت می رسد، سرانجام به حافظهٔ کلاستر دریافت می رسد .



شکل ۳-۱ : مسیر یک بسته بین دو گره محاسباتی

در زمان طراحی کردن بافت اتصال برای یک کلاستر کامپیوتری لینوکس انتخاب کارت های بین سطوحی شبکه و سوئیچ های مطابق آنها، اگرلازم باشد، به میزان زیادی به تقاضا ها برای چیزی که کلاستر ساخته می شود، بستگی دارد. پارامترهای اصلی نوع شبکه که این انتخاب را تحت تأثیر قرار می دهند عبارتند از : مرحله ی پنهانی[۱۳]، پهنای باند، سرجمع CPU [۱۴]، چند فهرستی[۱۵] والبته هزینه ی وسیله های شبکه ای و بسته های نرم افزاری مورد نیاز از این قبیل می باشند در بخش های بعدی، ما هر یک از این پارامتر ها را تعریف می کنیم و روش های مقایسه کردن انتخاب های متعدد را به همراه این پارامترها توصیف می کنیم.

---

[۱۳] Latency

[۱۴] CPU overhead

[۱۵] Multithreadinding

مرحله ی پنهانی

به زمانی برمی گردد که طول می کشد یک بسته ی تنها منبع را ترک می کند و به مقصد می رسد. آن از زمان فرایند ارسال کردن که بسته را می فرستد به زمان فرایند مقصد که دریافت بسته را شروع می کند اندازه گیری می شود.

یک روش مشهور از اندازه گیری کردن زمان از یک اتصال داشتن یک جفت از ماشین ها که به طور مکرر یک پیام کوچک را به جلو و عقب می فرستند، می باشد. اگر بسته قادر باشد گردش مسیر N را درزمان T انجام دهد، یک روش زمان از شبکه T/2N خواهد بود. اگر MPI درحال برنامه ریزی کردن رابط برنامه نویسی کاربردی(API)که مورد استفاده قرار می گیرد، باشد، این زمان اندازه گیری شده به عنوان زمان MPI گردش مسیر نیم فاصله برمی گردد.

اندازه گیری زمان یک بافت درسطح API مهم می باشد. حتی اگر رسانه ی فیزیکی زیر بنایی بتواند از مبادله ی خیلی سریع پیام ها حمایت کند، اجرای پروتکل روی رسانه ی فیزیکی می تواند باعث یک اختلاف مشخصی در زمان موثر برای تقاضا شود. برای مثال اگر کتابخانه ی برنامه نویسی موازی روی یک پروتکل هدف عمومی مانند TCP/IP اجرا شود، زمان مشخص طول می کشد تا بسته از طریق لایه های متعدد از الگو به خود رسانه ی فیزیکی برسد. دراجرائیات MPI روی داده های TCP/IP اولین مورد کپی شده از NIC به فضای هسته می باشد و سپس از فضای کرنل به فضای استفاده کننده می باشد. اگر چه لینوکس یک اجرای عالی از TCP/IP دارد، برای بسیاری از محیط های کاربردی MPI، استفاده از الگوی وزن سبک مورد مطلوب می باشد. همچنین مهم است که تأ ثیر سوئیچ ها را روی زمان بین دو ماشین درکلاستر را به حساب آوریم اگرپیغام ها مجبور باشند از طریق چند سویچ بروند قبل از اینکه به گره ی مقصد خودشان برسند، زمان برای این پیغام ها به طور قابل توجهی درمقایسه با گره های مرتبط نقطه به نقطه، بالاتر خواهد بود. چنین ساختار هایی همچنین تعداد زمان مختلف بین جفت هایی از گره ها معرفی می کنند.

پهنای باند

به میزان داده های اشاره می کنند که از طریق بافت اتصال در یک واحدی از زمان جریان می یابند. همانند زمان، پهنای باند نیز باید درسطح موازای API اندازه گیری شود، زیرا پهنای باند خام از شبکه ی فیزیکی می

تواند به طور قابل توجهی بزرگتر از چیزی که تقاضای موازی بتواند واقعاً تشخیص دهد، می باشد.

دو شماره از پهنای باند وجود دارند که نیاز است درنظر گرفته شوند. اولی پهنای باند شبکه همراه با ارتباط نقطه به نقطه بین دو گره ها در یک کلاستر می باشد. اگر چه درعمل یک کلاستر به ندرت دراین ساختار جریان می یابد، با این حال این شماره ی پهنای باند بهترین نمایش را می دهد که یک پورت تنها حمایت خواهد کرد. شماره ی دومی، پهنای باند دو قسمتی از بافت اتصال می باشد. به طور مفهومی، پهنای باند دو قسمتی، میزان اطلاعات است که می توانند از طریق یک خط مجازی که کلاستر را به دو نیم تقسیم می کند فرستاده شود و هر کدام از آنها با تعداد مساوی از CPUs می باشند. با وابسته بودن روی اینکه این خط مجازی چگونه تقسیم می شود، شماره های مختلف پهنای باند را می توان بدست آورد.

اگر همه ی گره های محاسبه ای کلاستر به یک سویچ اتصال تنهایی متصل شوند، آنگاه پهنای باند دو قسمتی کلاستر همانند آن سویچ می باشد. مثلاً می گوییم یک سویچ N پورت دارد (با فرض اینکه N زوج باشد). هم اکنون، فرض کنید که گره های N/2 درکلاستر درحال ارسال کردن یک جریان ممتدی از داده ها می باشند . به عبارت دیگر یک بسته ی خیلی بزرگ به گره های دیگر N/2 درکلاستر می فرستند. این دومین کلاستر از گره های N/2 نیز درحال ارسال کردن یک جریان ممتد برای برگشت به اولین کلاستر می باشد. (این بر آن فرض است که شبکه به طور کامل دوتایی باشد). میزان داده هایی که از طریق سویچ بر واحد زمان جریان می یابند، پهنای باند دو قسمتی را از سویچ مشخص می کنند. این بالاترین پهنای باند شبکه می باشدکه یک کلاستر می تواند انتظار آن را داشته باشد، اگر سویچ درنمایش بالا درمرکز بافت آن باشد. یک سویچ بدون مانع پهنای باند دو قسمتی مساوی با N برابر پهنای باند نقطه به نقطه دارد. پهنای باند از یک سویچ مورد قبول کمتر از سویچ مطابق بدون مانع با همان شماره ازپورت ها می باشد. بنابراین، یک سویچ مورد نظر ممکن است یک گره را از فرستادن یک پیام کاهش یا کُند بکند حتی اگر ارتباط باعث پهنای باند بالاتری شود. دریک کلاستر با اتصال خیلی شلوغ همراه با گره هایی که مجبورند داده ها را به طور همزمان درزمان های متعددی بفرستند، استفاده ازیک سویچ غیر مانع ضروری می باشد. دریک کلاستری که برای گره ها به ندرت اتفاق می افتد که داده ها را به طور همزمان به شبکه بفرستند، یک گزینه ی مورد قبول ممکن است یک انتخاب مقرون به صرفه باشد. یک شماره ی مهم برای توجه قرار دادن، پهنای باند دو قسمتی درپردازشگر درکلاستر می باشد. دوباره فرض می کنیم که همه ی گره ها به یک سویچ مرکزی تنهایی متصل می باشند، سویچ بدون مانع یک

پهنای باند ثابتی را در هر پردازشگر به عنوان شماره گره ها درکلاستر افزایش می دهد. ازطرف دیگر، یک سویچ مورد قبول ،پهنای باند را برای هرپردازشگر به یک نقطه مقیاس بندی می کند که افزایش درشماره ی گره ها ،پهنای باند موجود درپردازشگر را کاهش می دهد.

زمانی که تعداد گره ها دریک کلاستر بالاتر از پورت های موجود دریک گزینه ی تنها باشند، نگه داشتن پهنای باند ثابت در پردازشگر سخت تر می شود ..

یک روش واضح برای جلوگیری از این، استفاده کردن از سویچ ها با میزان بالاتری از پورت ها می باشد. این گاهی اوقات عملی نمی شود، زیرا به سرمایه ی زیادی در ظرفیت نیاز دارد که ممکن است در آینده مورد استفاده قرار بگیرد.علاوه بر آن برای کلاستر های بزرگتر، سویچ ها ممکن است وجود نداشته باشند تا احتیاجات پورت را برای همه ی گره ها برآورده سازد. روش دیگر برای جبران کردن حداقل بعضی از این باریک شدن، استفاده کردن از سویچ ها با ارتباط های سریع تر می باشند. اگر بیش از دو سویچ دریافت اتصال مورد نیاز باشد، یا اگر ارتباط سریع تر یک انتخاب نمی باشد، توپولوژیهای پیچیده ی بیشتری از ارتباط های داخل سویچ ای برای استفاده جهت کاهش بدترین نمایش از چندین جهش مورد نیاز می باشد. تقاضا های مواز ی زیادی وجود دارند که بار ارتباطی بالاتری را روی گره های انتخاب شده کمی قرار می دهند، به عبارت دیگر بار ارتباطی آنها از طریق کلاستر به طور زوج موازنه نشده است.

اگر یک کلاستر برای چنین کار بردهایی سازگاری پیدا کند، یک توپولوژی غیر منظم از ارتباط های شبکه می تواند مورد استفاده قرار بگیرد. برای مثال چندین ارتباط می توانند به یک گره ی تنها اضافه شوند. به طور عمومی ما استفاده کردن از چنین توپولوژیهای شبکه ای نامنظم را فقط درموارد ویژه ای توصیه می کنیم که کلاستری که به کار برده می شود برای راه اندازی کاربردهایی با نامتوازنی ارتباط برای بیشتر آرایش چیدمان آن باشد و توپولوژیهای منظم باعث آسان شدن مدیریت می شود و باعث می شود که از الگوریتم های طراحی کار و روش قابل فهمی استفاده شود، هما نطور که برای مدیر آسان می شود.

سرجمع سی پی یو

درحین بررسی کردن اینکه تقاضا چگونه برای انجام شدن روی یک کلاستر به پیش می رود، مهم است تأثیر فعالیت شبکه را روی عملکرد سی پی یو درنظر بگیریم. اگر حجم زیادی از پردازش شبکه ای در سی پی یو میزان اتفاق افتد سپس این بار با میزان دستور العمل ها در بین خود تقاضا رقابت خواهد کرد.

بسیاری از NIC های هوشمند سعی می کنند قطعات متعددی از پردازش شبکه ای را در بین خودشان به کار برند، بنابراین سی پی یو را برای انجام دادن کار تقاضا آزاد می گذارند. برای یک اتصال خیلی شلوغ این می تواند یک معیار اصلی باشد، که یا NIC نیاز است که هوشمند باشد، یا لازم است که قدرت سی پی یو برای تطبیق نمودن بار شبکه بالاتر باشد. یک سرجمع پایین سی پی یو و هم پوشی با دقت از محاسبات همراه ارتباطات، باعث می شود که تقاضا، کار مفید را درCPU انجام دهد درحین اینکه فرایند درپردازشگری که در NIC نهفته است انجام می گیرد .

NIC میرینت[۱۶] یک نمونه ای از چنین NIC های هوشمند می باشد. NIC میرینت یک پردازشگر ی دارد که روی کارت نهفته می باشد و به پردازشگر لانای[۲] معروف می باشد. پردازشگر لانای[۱۷] با ر مسئولیت سی پی یو میزبان را با به عهده گرفتن بسیاری از فعالیتهای جریان شبکه، کاهش می دهد.

بهینه سازی ویژه ی دیگر برای کاهش دادن سرجمع CPU، استفاده کردن از پروتکل های سبک وزن می باشد. این پروتکل ها به طور ویژه ای برای حمل کردن پیام های نسبتاً کوچک درطول یک توپولوژی شبکه همانند یک کلاستر می باشند . پروتکل های قدیم همانند TCP/IP برای انجام دادن هدف های عمومی شبکه طراحی می شوند و از این رو باید از بسیاری از خطا ها و استثنائات که با کلاستر های محاسباتی مرتبط می باشند، مواظبت کنند. پروتکل های استفاده شده دراتصال های کلاستر می توانند خوش بینانه تر باشند و می توانند فرضیات متعددی همانطور در مقایسه با پروتکل های هدف عمومی ایجاد کنند. اجرای چنین پروتکل هایی عموماً سطحی ازهوش را از NIC اساسی می طلبد. این می تواند یا درشکل پردازشگر نهفته شده باشد، مانند پرداز شگر لانای دریک NIC میرینت، یا آن می تواند درشکل نرم افزار ثابت ویژه ای درNIC باشد.

---

[۱۶] Myrint

[۱۷] Lany

Multithreading:

به توانایی شبکه برای انتقال دادن چندین پیغام به طور همزمان همانند توانایی یک گره برای استفاده از بافتی بین چندین بافت، اشاره می کند.

دریک کلاستر محاسباتی، استفاده کردن از گره های SMP به عبارتی دیگر گره هایی که حاوی بیش از یک پردازشگر می باشند،کارها ممکن است دریک روشی زمان بندی شوند که تقاضا های مستقل روی همان گره می خواهند اتصال را برا ی مبادله ی پیغام آنها استفاده کنند .

NIC های ویژه (و درایور های های مطابق آنها) قادرند برای پردازش های سطح بالا چندتایی برای مبادله ی پیغام صحبت کنند، درحالی که بعضی از NIC ها به یک فرایندی متصل می شوند تا زمانی که ارتباط خودش را کامل کرده باشد و NIC را ترک کند.

هزینه

درحین محاسبه کردن هزینه ی یک اتصال، چندین عوامل باید درنظر گرفته شوند. هزینه اصلی از اتصال شامل هزینه ی NIC ها ، سویچ ها، کابل ها وکتابخانه های نرم افزاری مورد نیاز می باشد. ملاحضات هزینه ی ثانویه نیز وجود دارند. انتخاب ویژه از NIC ممکن است به پردازشگرهای میزبان با سرعت بالاتر نیاز داشته باشد و بنابراین به هزینه اضافه می شود. یک شبکه ی غیر استاندارد همچنین به هزینه ی آموزش نیاز دارد، که ممکن است لازم نباشد اگر یک شبکه استفاده شده ای به طور معمولی همانند اترنت مورد استفاده قرار بگیرد. موجود بودن درایور خوب و اجرائیات خوب API روی انتخاب های شبکه ی فروش ویژه ممکن است هزینه های پیشرفت را کاهش دهند.

اتصال های مشهور

این بخش انتخاب های اتصال متعدد را به طورکنونی یا موجود در بازار و تجارت های متعدد درطول خطوطی از پارامترهای بالا توصیف می کند.

اترنت سریع[۱۸] واترنت گیگا بیت[۱۹]

انواع متعدد ی از اترنت، پروتکل های TCP/IP را از موتور مرکزی اینترانت و اینترنت اجرا می کند. هم اکنون سه نوع مشهور از اترنت که درپهنای باند خودشان متفاوت می باشند، وجود دارد: اترنت استاندارد، اترنت سریع و اترنت گیگا بیت که به ترتیبMbps ۱۰، Mbps ۱۰۰ و Mbps ۱۰۰۰ از پهنای باند تئوریکی فراهم می کند. اکثر دسک تاپ ها امروزه با استفاده از اترنت به شبکه های ناحیه ی محلی خودشان مرتبط می شوند. اترنت سریع خیلی زود بدست می آورد، و مکانیسم ارتباطی برجسته ای برای سرورهای پشت خط جدید می باشد و همچنین برای سرور های وب و فایل سرورها. اترنت گیگا بیت درحال بدست آوردن سریع و پذیرش آن همانند ستون فقرات شبکه که سویچ ها را به همدیگر مرتبط می کند، می باشد، همانند یک ارتباط در سرورهای پشت خط که به طور سنگینی دسترسی پیدا کرده اند. عضوهای متوالی از خانواده ی اترنت به میزان زیادی از پایگاه نصب نسل قبلی برتری می یابند. اترنت گیگا بیت روی مس می تواند از همان کابل(CAT-5 USING 1000 BASE-T) استفاده کند همانطور که برای اترنت سریع براساس شبکه ها مورد استفاده قرار گرفته است. بالا بردن یک سرور از اترنت سریع به اترنت گیگا بیت به تغییر درتوپولوژی شبکه نیاز ی ندارد. سازمان ها می توانند روی ساختار زیر بنایی کنونی خودشان، ابزارها ی مدیریت شبکه، و پرسنل برای افزودن در پهنای باند مورد نیاز در هزینه های متوسط سطح بندی شوند. این عوامل باعث شده اند که اترنت و اترنت سریع مؤلفه های خانگی زیاد ی شوند. با پایین بودن هزینه های انواع متعددی از آنها، به عنوان بافت اتصال کلاستر مورد استفاده قرار گرفته اند. اگر چه اترنت استاندارد هنوز یک پورت خیلی مشهوری روی دسکتاپ ها می باشد، آن به عنوان یک انتخاب شبکه ای سرور کنار گرفته شده است. حتی اکثر سرورهای پایین با یک پورت اترنت سریع حمل می شوند.اگرچه سویچ های اترنت سریع ازروی قیمت درحال پایین آمدن می باشند، یک هاب تکرار کننده ی ارزانتر از یک سویچ می باشد. برای کلاستر های که پهنای باند دو قسمتی و قابلیت مقیاس پذیری کلاستر اهمیتی ندارد، استفاده از یک هاب می تواند در نظر گرفته شود. این عمل برای مواردی که تقاضاهای موازی روی کلاستر باعث می شوند که مبادله های کوچک از اطلاعات جریان داشته باشند نیز درست می باشد.

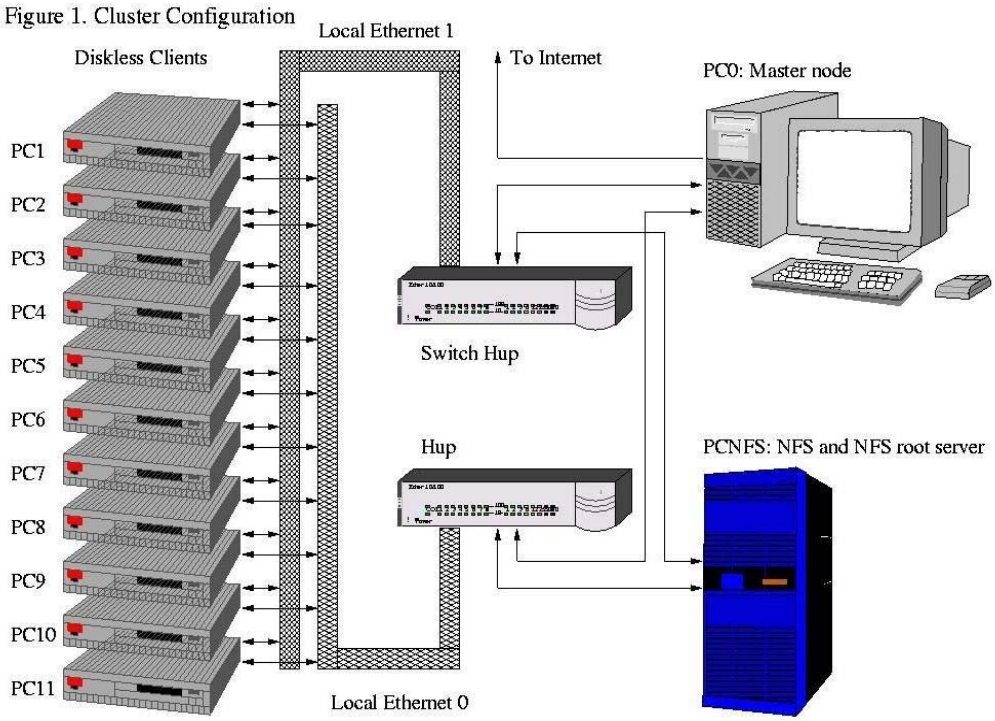
---

[۱۸] Fast Ethernet

[۱۹] Gigabit ethernet

فصل چهارم- چگونگی ساخت یک کلاستر محاسبه ای موازی

در این بخش ما قصد داریم روال ساخت یک کلاستر محاسبه ا ی موازی بدون دیسک را که برای محاسبات فیزیکی طراحی شده است توصیف کنیم. کلاستر شامل یک نود اصلی، یک نود سرور NFS -NFS وROOT و چندین نود کلاینت بدون دیسک است. این نودها برای دست یافتن به توانایی محاسبه موازی به وسیله ی اترنت به هاب ها متصل هستند. پیکر بندی سخت افزار سیستم در شکل ۴-۱ نشان داده شده است:



شکل ۴-۱: پیکربندی سخت افزار کلاستر

جزئیات مشخصات کلاستر در جدول های زیر مشخص شده است:

۱.۴. مشخصات سیستم

۱.۱.۴.پیکربندی سخت افزار

یک نود اصلی:

جدول ۴-۱:مشخصات نود اصلی کلاستر

سخت افزار	تعداد	توصیف
CPU	۲	Pentium III 1GH
RAM	۱	512 M
ETHERNET CARD	۳	3c905c
HARD DISK	۱	30G
FLOPPY	۱	-----
VGA CARD	۱	-----
MONITOR	۱	-----

یک نود سرور سیستم فایل NFS و ROOT- NFS:

جدول ۴-۲:مشخصات نود سرور سیستم فایل NFSو ROOT– NFS

سخت افزار	تعداد	توصیف
CPU	۲	Pentium III 1GH
RAM	۱	512 M
ETHERNET CARD	۲	3c905c
HARD DISK	۱	30G
FLOPPY	۱	-----
VGA CARD	۱	-----
MONITOR	۱	-----

۱۱نود کلاینت :

جدول ۴-۳:مشخصات نود کلاینت

سخت افزار	تعداد	توصیف
CPU	۲	Pentium III 1GH
RAM	۱	512 M
ETHERNET CARD	۲	3c905c
FLOPPY	۱	------
VGA CARD	۱	-------

هاب :

جدول۴-۴:مشخصات هاب

سخت افزار	تعداد	توصیف
HUB	۱	D-Link DES-1016R
HUB	۱	D-Link DFE-916DX

۲.۱.۴.پیکربندی نرم افزار

سیستم عامل: RedHat 6.2

راه اندار شبکه: etherboot4.0

سیستم برنامه نویسی موازی:mpich1.2.1

قبل از ارائه ی روال ساخت کلاستر، ابتدا خلاصه ای از مفاهیم سیستم عامل لینوکس را بیان می کنیم. بر اساس این اطلاعات، فهم بهتری از تمام مراحل روال ساخت خواهید داشت.

۲.۴.سیستم عامل لینوکس

تلاش برای ساخت کلاستر مورد نظر اساساً روی چگونگی راه اندازی هسته و چگونگی نصب سیستم فایل

ریشه از یک سرور دور است. به دلیل اینکه نود کلاینت برای میزبانی سیستم فایل و هسته ی خودش یک هارد دیسک ندارد. بنابراین سیستم فایل و هسته ی کلاینت ها باید به وسیله ی سرور دیگر از طریق اتصالات شبکه ای ایجاد شوند.

- نود سرور یک نصب کامل از سیستم عامل لینوکسRedHat را نیاز دارد .
- سرور باید یک تصویرهسته ی ساده شده ای را برای نود های کلاینت برای دانلود آن آماده کند .
- سرور باید یک حداقل سیستم فایل ریشه را برای نود کلاینت برای نصب آن آماده نماید .
- نود کلاینت باید یک دیسک راه انداز شبکه برای راه اندازی از درایور فلاپی خود داشته باشد، سپس تصویر هسته را از شبکه در خواست نماید.
- نود کلاینت باید یک تصویر هسته از سرور بدست آورده آن را دانلود نماید و غیر فشرده کرده و آن را اجرا کند .
- تصویر هسته برای کلاینت سیستم فایل ریشه ی خودش را به عنوان سیستم فایل ریشه ی NFS نصب خواهد کرد بر اساس تحلیل بالا ما نیازمند به داشتن اطلاعاتی راجع به موارد زیر هستیم:

هسته ی لینوکس چیست؟

ساخت یک تصویر هسته برای کلاینت بدون دیسک

ساخت یک سیستم فایل ریشه برای کلاینت بدون دیسک

روال راه اندازی لینوکس

راه اندازی هسته ی سرور

راه اندازی هسته ی اصلی کلاینت

۱.۲.۴.هسته ی لینوکس چیست؟

هدف هسته ی لینوکس جداکردن پیچیدگی سخت افزار از کاربران است. هسته یک مجموعه از توابع سیستمی را فراهم می کند که برای جلوگیری از سر و کار داشتن کاربران با سخت افزار فراخوانی می شوند. به

عنوان مثال اگر کاربران می خواهند به یک فایل از هارد دیسک دسترسی یابند حقیقتاً یک سیستم خواندن صادر می شود که هسته را فراخوانی می کند و هسته با جزئیاتی مثل حرکت بازوی خواندن/نوشتن دیسک به محل درست از هارد دیسک سرو کار خواهد داشت و بخش هایی از فایل را به کاربران بر می گرداند .

هسته ی لینوکس یک سیستم چند کاربره و چند فرایندی است. آن شامل چندین بخش مانند : مدیریت فرایند، مدیریت حافظه، سیستم فایل ها، کنترل ابزار های شبکه بندی و غیره است.آن به درخواست های کاربران با تخصیص CPU، حافظه، دستگاه های ورودی و خروجی در یک راه نسبتاً خوب پاسخ می دهد. اگر سیستم بخواهد کار کند اولین چیز دانلود و اجرای هسته ی لینوکس است. اندازه ی هسته متناسب با تقاضا روی هم رفته می تواند بزرگ یا کوچک باشد. در حالت عادی هسته ی بزرگتر سرویس زیادی را تولید می کند اما CPU زمان زیادی مصرف می کند که باعث می شود سیستم به مرور زمان از کار بیفتد.

۲.۲.۴.ساخت تصویری از هسته

نصب هسته :

- اگر شما منابع کامل را نصب می کنید دستور زیر را انجام دهید:

- cd /usr/src
- gzip -cd linux-2.2.XX.tar.gz | tar xvf -
- با انجام دستورات زیر مطمئن باشید که شما هیچ فایل قدیمی و کهنه ای ندارید :

- cd /usr/src/linux
- make mrproper

پیکربندی هسته

- دستور make config را برای پیکربندی هسته انجام دهید. دستور make config برای کار کردن BASH را نیاز دارد .

- دستورات متناوب پیکربندی زیر زا به کار گیرید :

- make menuconfig
- make xconfig
- make oldconfig

- سرانجام انجام دهید make dep برای تنظیم همه ی وابستگی ها به طور صحیح و درست

گردآوری هسته :

دستور make zImage یا make bzImage برای ایجاد یک تصویر هسته ی فشرده شده انجام دهید .

اگر شما هر یک از بخش های هسته را پیکربندی کردید مثل modules شما باید دستور make modules با پیروی کردن از make modules_install انجام دهید.

۳.۲.۴.ساخت یک سیستم فایل ریشه

درکنار هسته، هم چنین شما یک سیستم فایل ریشه برای میزبانی برنامه ها ، پیکربندی ها و داده ها نیاز دارید. ساخت سیستم فایل ریشه شامل انتخاب فایل های لازم برای سیستم، برای اجرا کردن آنهاست. یک سیستم فایل ریشه باید تمام چیزهای مورد نیاز برای سیستم لینوکس را در بر بگیرد. برای داشتن توانایی برای انجام این کار دیسک باید شامل کمترین نیازها برای یک سیستم لینوکس باشد .

ساختار سیستم فایل اصلی

مجموعه ای از دایرکتوری ها : /dev ، /proc ، /bin ، /etc ، /lib ، /usr، /tmp

مجموعه ی اصلی از امکانات: sh، ls ، cp ، mv، etc

حداقل مجموعه از فایل ها پیکربندی : rc، inittab، fstab، etc

دستگاه ها: /dev/hd* ، /dev/tty*، /dev/fd0

کتابخانه ی زمان اجرا برای تولید تابع های ابتدایی استفاده شده با امکانات

به عبارت دیگر، برای ساخت یک سیستم فایل ریشه شما یک دستگاه جداگانه که به اندازه ی کافی بزرگ است را برای نگهداری همه ی فایل ها قبل از فشرده شدن نیاز دارید. چندین انتخاب وجود دارد که ما ramdisk را انتخاب می کنیم .

یک ramdisk (DEVICE=/dev/ram) استفاده کنید و

DEVICE را با دستور زیر آماده کنید:

```
dd if=/dev/zero of=/dev/ram0 bs=1k count=4096
```

سپس، فایل سیستم را با انجام دستور خط زیر بسازید :

```
mke2fs -m 0 -N 2000 /dev/ram0
```

سپس،یک نقطه ی اتصال بسازید و دستگاه را متصل کنید:

```
mkdir -p /tmp/ramdisk
mount -t ext2 /dev/ram0 /tmp/ramdisk
```

اینجا یک حداقل مجموعه ی معقول از دایرکتوری سیستم فایل ریشه وجود دارد :

dev/ فایل های ابزارهای نیازمند برای اجرای ورودی/خروجی

proc/دایرکتوری ریشه برای سیستم فایل

etc/فایل های پیکربندی سیستم

sbin/فایل های سیستم بحرانی

bin/بخش مطرح شده برای فایل های باینری اصلی از یک سیستم

mnt/یک نقطه ی پایه برای نگهداری روی دیسک های دیگر

usr/تقاضاها و برنامه های اضافی

ابتدا دایرکتوری های فهرست شده در بالا را ایجاد کنید.

```
Cd /tmp/ramdisk
mkdir dev proc etc sbin bin mnt usr usr/lib
```

برای ایجادdev/

```
cp -dpR /dev/fd[01]* /tmp/ramdisk/dev
cp -dpR /dev/tty[0-6] /tmp/ramdisk/dev
```

را استفاده نمایید.

سرانجام پس از تنظیم کتابخانه ها شما نیاز دارید دستور ldconfig را برای انتخاب دوباره ی /etc/ld.so.cache روی سیستم فایل ریشه اجرا کنید. حافظه ی کش به بار کننده اعلام می کند که کتابخانه ها را پیدا کند. شما می توانید این کار را با دستور زیر انجام دهید:

```
ldconfig -r /tmp/ramdisk
```

وقتی که شما ساخت سیستم فایل ریشه را تمام کردید آن را به یک فایل کپی کنید و به هم فشرده کنید:

```
unmount /tmp/ramdisk
dd if=/dev/ram0 bs=1k | gzip -v9 > rootfs.gz
```

انتقال سیستم فایل ریشه

```
dd if=rootfs.gz of=/dev/fd0 bs=1k seek=KERNEL_BLOCK
```

۴.۲.۴.روال راه اندازی سیستم عامل لینوکس

همه ی سیستم های PC فرایند راه اندازی را با اجرای کد در حافظه ی ROM (مخصوصاً BIOS) برای بارگذاری بخشی از سکتور۰ و سیلندر۰ درایو راه انداز شروع می کنند. درایو راه انداز معمولاً اولین درایو فلاپی (dev/fd0) یا اولین هارد دیسک (dev/hda) می باشد. سپس بایوس سعی می کند این سکتور را اجرا کند. وقتی که هسته کاملاً بارگذاری شد ،آن درایور ها ی دستگاه و ساختارها ی داده ی داخلی خود را مقدار دهی اولیه می کند .

۵.۴.جزئیات روال ساخت کلاستر کامپیوتر های شخصی بدون دیسک

روال ساخت یک کلاستر PC می تواند به دو بخش تقسیم شود:

تنظیمات سرور NFS-ROOT وNFS

تنظیم گره های کلاستر

تنظیمات سرور NFS-ROOT وNFS

چندین راه برای نصب سرور وجود دارد. یا به وسیله ی دیسک سخت و یا به وسیله ی شبکه. برای سادگی اجازه بدهید فرض کنیم که شما یک نصب کامل از سیستم عامل RedHat 6.2 را روی سرورر خود دارید. اگر روال نصب صحیح انجام شود شما باید یک سیستم عامل کاملا تابعی را با ارتباطات شبکه ای داشته باشید. سپس شما باید یک دیسک راه ا نداز شبکه را برای کامپیوتر های گره آماده کنید از طریق مراحلی که در زیر آمده :

راه اندازی شبکه از یک فلاپی

etherboot 4.0 و etherboot 4.7.24 را دانلود کنید. فایل floppyload.bin را از etherboot4.0/bin و فایل 3c905c-tpo.lzrom را از etherboot-4.7.24/src/bin32 بگیرید. سپس دستور زیر را

```
# cat floppyload.bin 3c905c-tpo.lzrom > /dev/fd0
```

برای ایجاد یک فلاپی راه انداز از شبکه وارد کنید.

روال تنظیمات سرور dhcp

فایل /etc/dhcpd.conf را مانند نمونه آماده کنید:

```
# Sample configuration file for ISCD dhcpd
#
# Don't forget to set run_dhcpd=1 in /etc/init.d/dhcpd
# once you adjusted this file and copied it to /etc/dhcpd.conf.
#

default-lease-time          21600;
max-lease-time              21600;

option subnet-mask          255.255.255.0;
option broadcast-address    192.168.0.255;

shared-network WORKSTATIONS {
   subnet 192.168.0.0 netmask 255.255.255.0 {
   }
}
group  {
   use-host-decl-names      on;
   option log-servers       192.168.0.254;

   host pc1 {
```

```
•
•       hardware ethernet    00:01:02:92:70:69;
•       fixed-address        192.168.0.1;
•       filename             "/tftpboot/pc1/vmlinuz.3c905nomodPc1";
•    }
•    host pc2 {
•       hardware ethernet    00:01:02:91:43:0F;
•       fixed-address        192.168.0.2;
•       filename             "/tftpboot/pc2/vmlinuz.3c905nomodPc2";
•    }
•    host pc3 {
•       hardware ethernet    00:01:02:92:70:18;
•       fixed-address        192.168.0.3;
•       filename             "/tftpboot/pc3/vmlinuz.3c905nomodPc3";
•    }
•    host pc4 {
•       hardware ethernet    00:01:02:91:43:45;
•       fixed-address        192.168.0.4;
•       filename             "/tftpboot/pc4/vmlinuz.3c905nomodPc4";
•    }
• }
```

فایل متنی /etc/rc.d/init.d/dhcpd را ویرایش کنید و خط daemon /usr/sbin/dhcpd را پیدا کنید.سپس آن را به daemon /usr/sbin/dhcpd eth1 تغییر دهید. بررسی نمایید که آیا فایل /var/state/dhcp/dhcpd.leases وجود دارد یا نه در غیر این صورت آن را ایجاد نمایید.یک اتصال نرمی را در /etc/rc.d/rc3.d اضافه کنید.حالا شما می توانید سرور dhcp را با گذاشتن فلاپی netboot درگره ی pc و روشن کردن آن تست کنید.

روال تنظیمات سرور ftp

/etc/services را برا ی اینکه مطمئن شوید خط

```
tftp          69/udp
```

وجود دارد بررسی نمایید و /etc/inetd.conf را برای اینکه مطمئن شوید خط زیر :

```
tftp   dgram   udp    wait    root    /usr/sbin/tcpd  in.tftpd
```

غیر توضیحی است.

خواندن فایل های پیکر بندی را برای فهم دوباره شروع کنید.

شما مطمئن باشید که /etc/hosts.allow خط .ALL: 192.168.0 را در بر دارد. نام میزبان را در /etc/hosts/ اضافه کنید و باید شامل بخش هایی از /etc/dhcpd.conf/ باشد.

دایرکتوری ریشه ی گره های pc1،pc2، pc3 وpc4 را روی دایرکتوری /tftpboot سرورتان با دستورات زیر ایجاد نمایید:

- mkdir -p /tftpboot/pc1
- mkdir -p /tftpboot/pc2
- mkdir -p /tftpboot/pc3
- mkdir -p /tftpboot/pc4

هسته را برای گره ی کلاینت آماده کنید. وقتی که پارامتر های هسته را تنظیم می کنید شما باید مطمئن باشید که بخش های زیر را مشخص کرده اید:

- * No module support (for simplicity).
- * Support for your specific network card, for example, 3com 3c905c.
- * RAM disk support.
- * BOOTP support.
- * /proc filesystem support.
- * NFS filesystem support.
- * Root file system on NFS

بعد از اینکه که شما یک نام جدید از منبع هسته ایجاد کردید دستورا ت زیر را برای ساخت یک kernel network bootable انجام دهید :

- /mknbi-linux --rootdir=/tftpboot/pc$1/pc$1root
- /usr/src/linux/arch/i386/boot/bzImage >
- /tftpboot/pc$1/vmlinuz.3c905nomodPc$1

سیستم فایل ریشه را برای هر کلاینت آماده کنیدسیستم فایل ریشه را به /tftpboot/pc1کپی کنید تنظیمات شبکه و تنظیمات NFS را در دایرکتوری /tftpboot/pc1/etc تغییر دهید.

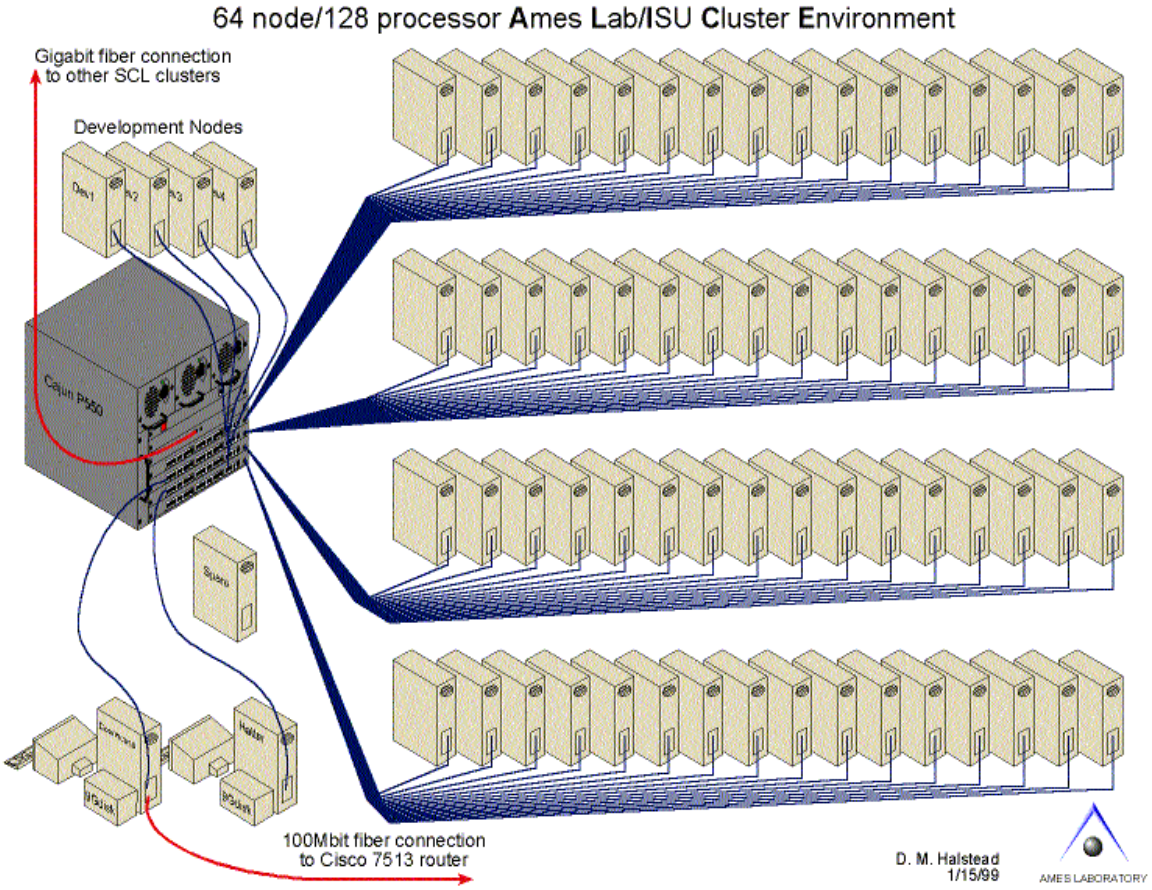
فصل پنجم - کلاستر های نمونه

کلاستر ALICE

ALICE *(Alpha-Linux-Cluster-Engine)* ، یک کلاستر با ۶۴ پردازشگر دوگانه (۱۲۸ پردازشگر Pentium *Alpha*) است که در ۲۰۰ مگاهرتز اجرا می شوند. همه ی کامپیوتر ها با یک سویچ اترنت مرکزی به هم متصل هستند که یک توپولوژی مسطح تولید می شود. به علاوه یک گره ی اصلی، یک فایل سرور و چهار گره ی گسترش وجود دارد .

زمانبند PBS همه ی کارها را کنترل می کند . در این کلاستر کامپایلر های گروه Portland استفاده می شود، که کد تولیدی توسط آنها۵۰ ٪ سریعتر از کامپایلر هایg۷۷ هستند .

شکل زیر معماری این کلاستر را نشان می دهد :



ALICE

شکل۵-۱:پیکربندی سخت افزار کلاستر ALICE

اتصال گره ها از نوع ParaStation است . ParaStation یک کتابخانه ی ارتباطات بسیار پایدار و بهینه را برای MPI (Message Passing Interface) و برای فرترن، C و C++ ارائه می دهد و یک latency بسیار کم دارد. حقیقت دیگر ی که در استفاده از این نوع اتصال خیلی مهم است از نظر قابلیت استفاده و قابلیت اطمینان است: درحالت موازی برای اصلاح خطای درون بسته های مخابره شده ی خودش،ParaStation واقعاً اطمینان می دهد که بستها به گره ی دریافتی می رسند، اگر بسته ها از بین بروند به صورت اتوماتیک دوباره به مقصد می فرستد .

منابع:

[1]Linux Documentation Project (http://www.linuxdoc.org)

[2]The Beowulf Project (http://www.beowulf.org)

[3] Hai Jin, Rajkumar Buyya, Mark Baker. Cluster Computing Tools, Applications, and Australian Initiatives for Low Cost Supercomputing

[4] Logan G. Harbaugh. Building High-Performance Linux Clusters, Sponsored by Appro, JUNE 2004

[5]Turbolinux(http://www.turbolinux.com/products/middleware/lb10/docs/user_guide/introduction.html)

[6]ExforsysInc(http://www.turbolinux.com/products/middleware/lb10/docs/user_guide/introduction.html)

Arak University

Faculty of Engineering

Department of Computer Engineering

B.Sc Thesis

Hardware Systems Significative And Actual Model Designing of Parallel Processing

Supervisor

Dr.Moghimi

BY: Nasrin Ghasemi

Nov 2009