جلسه هفتم

سیستم های توزیع شده (Distributed operating systemes)

در ابتدا تعاریف مطرح شده در ارتباط با یک سیستم توزیع شده و جنبه های مهم آن بررسی خواهد شد. سپس اهداف و خواصی که یک سیستم توزیع شده باید داشته باشد مورد بررسی قرار خواهد گرفت.

یک سیستم توزیع شده مجموعه‌ای از کامپیوترهای مستقل است که برای کاربران به عنوان یک سیستم هماهنگ نمایش داده می شود. به عبارت دیگر کاربرانی که از یک سیستم توزیع شده استفاده می کنند، هیچ گاه متوجه نخواهند شد، یا به عبارتی دیگر نباید متوجه شوند که در زیر مجموعه ی این سیستم بیش از یک کامپیوتر وجود دارد. معمولاً هر سیستمی از دو جنبه ی اصلی قابل بررسی است و این قاعده در مورد سیستم های توزیع شده نیز مستثنی نیست، لذا یک سیستم توزیع شده از دو جنبه ی سخت افزاری و نرم افزاری بررسی می شود.

از دیدگاه سخت افزاری یکی از مهم ترین نکاتی که در یک سیستم توزیع شده مهم است این است که ماشین‌هایی که در این سیستم وجود دارند، در اصطلاح Autonomous یا خود تصمیم پذیر هستند. در واقع هر ماشینی می تواند مستقل از ماشین های دیگر تصمیم بگیرد، در عین حال که با ماشین های دیگر همگام است.

از دیدگاه نرم افزاری، باید نرم افزار طراحی شده در اینگونه سیستم ها به گونه ای باشد که کاربر تصور کند که تنها با یک ماشین واحد صحبت می کند.

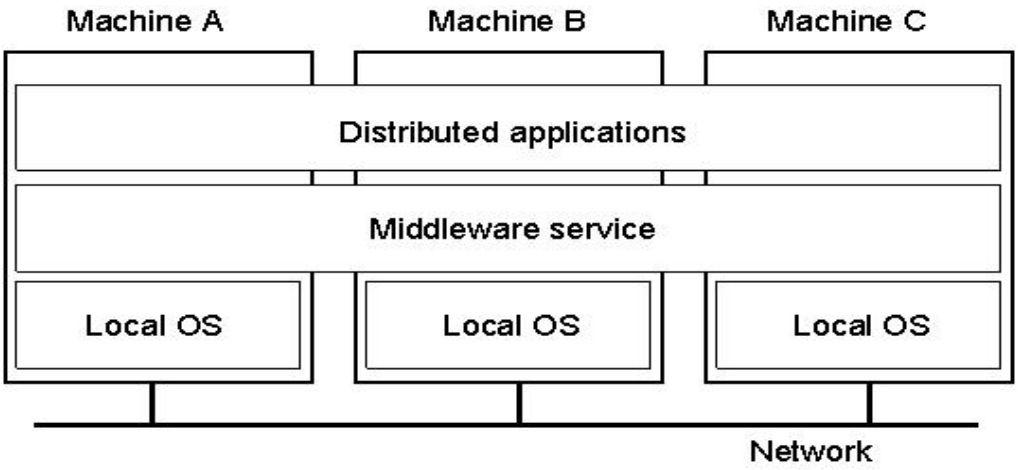
خواص سیستم های توزیع شده

اول اینکه در یک سیستم توزیع شده تفاوت های بین ماشین ها و نحوه ارتباط با آنها باید از کاربر پوشیده باشد. دوم اینکه این سیستم ها باید به گونه ای طراحی شوند که قابلیت گسترش داشته باشند، به عبارت دیگر رشد آنها به سادگی انجام گیرد. سوم اینکه لحظه ی برقراری ارتباط با اینگونه سیستم ها از یک روش یکنواخت و سازگار تبعیت کند. چهارم اینکه اینگونه سیستم ها باید به گونه ای طراحی شوند که از دیدگاه کاربر همواره موجود باشند و هیچ گاه کاربر احساس نکند قابل دسترسی نیست.

از جمله تعاریف دیگری که برای سیستم‌های توزیع شده عنوان می‌شود، تعریفی است که غالباً در سیستم های غیرهمگن مورد بررسی قرار می گیرد. ذکر این نکته ضروری است که سیستم های توزیع شده به دو گروه همگن و غیرهمگن تقسیم می شوند. سیستم های همگن، سیستم هایی هستند که تمامی ماشین های موجود در سیستم توزیع شده از یک جنس هستند و قابلیت های مشابهی دارند. اما سیستم های غیرهمگن می توانند از ماشین های متفاوتی تشکیل شده باشند.

برای پشتیبانی یک سیستم غیرهمگن از یک دیدگاه واحد می توان تعریف زیر را ارائه کرد:

در این تعریف عنوان می‌شود که یک سیستم توزیع شده به عنوان یک Middleware یا میان افزار طراحی می‌شود. به گونه‌ای که لایه ی مربوط به میان افزار در این سیستم ها بر روی همه ماشین ها قرار گرفته است. در شکل، نمونه‌ی مدلینگ این سیستم ها، مشاهده می‌شود.



با توجه به شکل، چندین ماشین مختلف وجود دارد که از سیستم های عامل محلی خود برخوردارند، و لایه ی Middleware که همان سیستم توزیع شده است، بر روی آن قرار گرفته است و وظیفه ی مدیریت این سیستم ها و هماهنگی آنها را بر عهده دارد و سپس لایه ی Application های توزیع شده بر روی آن قرار می گیرد.

در ادامه می خواهیم با جزئیات بیشتری به اهداف مطرح، در یک سیستم توزیع شده بپردازیم.

اهداف:

یکی از اهداف سیستم های توزیع شده اتصال کاربرها و منابع به یکدیگر است. یعنی یک کاربر بتواند از منبعی که روی ماشین دیگری وجود دارد استفاده کند. قطعاً در این ارتباط امنیت بسیار مهم است. مفهوم بسیار مهم بعدی در سیستم های توزیع شده مبحث شفافیت سیستم ها است که تا به حال در تعاریف مطرح شده برای اینگونه سیستم ها به طور ضمنی به آن اشاره شده است. معنای شفافیت این است که فرایندها و منابعی که توزیع شده اند از دید کاربران مخفی باشند، یعنی چه بسا کاربری که بر روی یک ماشین کار می کند از منبعی

که بر روی ماشین دیگر است استفاده کند، اما سیستم باید به گونه ای Transparent طراحی شده باشد که این کاربر هیچ گاه این چنین حقیقتی را نداند.

هدف سوم در این سیستم ها باز بودن (Openness) سیستم ها است. یک Distributed system باز یا Open DS به این شکل تعریف می شود که می گوییم سیستمی است که سرویس هایی را بر اساس قواعد مشخص و استاندارد ارائه می کند که Semantic و Syntax این سرویس ها از پیش تعریف شده و قابل دسترسی است. بنابراین تمامی افرادی که این Semantic و syntax را می‌دانند می‌توانند به این سادگی از این سرویس ها استفاده کنند، به شرط آنکه سطح دسترسی لازم برای این کار را داشته باشند.

آخرین هدف در یک سیستم توزیع شده قابلیت گسترش یا بزرگ شدن سیستم است و از سه جنبه مورد بررسی قرار می‌گیرد. از دیدگاه اندازه، از دیدگاه جغرافیا و از دیدگاه سرپرستی(Administrative) که هر یک از این اهداف را از جنبه های مختلف آن بررسی خواهیم کرد.

شفافیت (Transparency)

با توجه به مطالبی که در قبل عنوان شد، مفهوم شفافیت این است که جزئیات موجود در سیستم توزیع شده از دید کاربر پنهان باشد و در واقع کاربر نداند که چه اتفاقی در لایه پایینی می افتد و هر کاربری کل سیستم توزیع شده را به شکل یک ماشین واحد و در اختیار خود ببیند. بدیهی است که برای ایجاد شفافیت(Transparency) موازنه ای بین کارایی (Performance) و شفافیت (Transparency) وجود دارد و با افزایش سطح شفافیت چه بسا که کارایی سیستم کاهش یابد. شفافیت سیستم از جنبه های مختلفی قابل تعریف و استفاده است که مورد بررسی قرار خواهد گرفت.

۱- Transparency از دیدگاه دسترسی

در واقع در این نوع Transparency عنوان می‌شود که تفاوتهای موجود در نحوه نمایش داده ها و نحوه استفاده از منابع، از دید کاربر پنهان می شود. در واقع کاربر نخواهد دانست که یک منبع چگونه مورد دسترسی قرار می گیرد، و تنها آن منبع را استفاده می کند.

۲- Transparency مکانی یا (Location Transparency)

در این نوع Transparency می‌گوییم که مکان قرار گرفتن یک منبع مخفی می شود و کاربر از آن اطلاعی ندارد.

۳- Transparency مهاجرت یا (Migration)

قرار گرفتن حرکت یک منبع از یک ماشین به ماشین دیگر را از دید کاربر مخفی می کند.

در مباحث بعدی عنوان خواهد شد که برای افزایش کارایی در سیستم های توزیع شده این اتفاق خواهد افتاد که یک منبع از یک ماشین جدا شده و به ماشین دیگری منتقل گردد. همین قضیه در مورد فرایندها نیز صادق است و این اتفاق را به عنوان مهاجرت خواهیم شناخت.

٤- Relocation Transparency یا بازجایگزینی

این مسأله را که یک Resource ممکن است به مکان دیگری در هنگامی که در حال استفاده است جابجا شود را از دید کاربرپنهان خواهد کرد.

۵- Replication Transparency یا تکرار

خواهیم دید که بعداً برای افزایش Availability یا وجود یک سیستم در دوره زمان، یکی از راههای افزایش اطمینان استفاده از Replication یا تکرار منابع است. بنابراین سیستم باید به گونه ای طراحی شود که اگر منابع تکرار شدند آنگاه یک منبع توسط چندین فرآیند ممکن است Share شود و این فرآیندها و یا این

کاربران برای دسترسی به این منابع رقابت کنند. در این حالت کاربران نباید این حس را داشته باشند و نباید بدانند که چند نفر همزمان بر روی یک منبع هستند و منتظر آن می باشند.

۶- Transparency از نوع Concurrency یا همروندی

این نوع Transparency، حالت دیگری است که باز هم به منابع مشترک می پردازد و در واقع در اینجا این که چندین کاربر همزمان از یک منبع استفاده می‌کنند باید از دید کاربران مخفی بماند. تفاوت بسیار اندکی بین Replication و Concurrency وجود دارد. در حالت اول کاربران نباید بدانند که کاربران دیگری به منبع دسترسی دارند در حالت دوم کاربران نباید بدانند که کاربران دیگری به طور همزمان در حال دسترسی به منبع هستند.

۷-نوع دیگر Transparency بحث Failure یا خطاست.

در این حالت مفاهیم مربوط به خطا و بازیافت خطا در یک منبع را مورد توجه قرار می دهد و مدعی است که کاربران نباید اطلاعاتی از این اتفاقات داشته باشند. به عنوان مثال چنانچه شما در حال استفاده از یک سیستم توزیع شده که شامل ده‌ها کامپیوتر دیگر است باشید و به ناگاه کامپیوتری که شما بر روی آن هستید و اطلاعات آن را مورد استفاده قرار می دهید، دچار مشکلی می‌شود بهتر است که سیستم به گونه ای هوشمندانه و سریع عمل نموده و منابع مربوطه را منتقل کند که شما به عنوان یک کاربر هیچ گاه متوجه چنین خطایی در سیستم نشوید. به هر حال این موارد اهداف مهمی هستند که شاید دسترسی به برخی از آنها به سادگی امکان پذیر نباشد.

۸- Transparency از نوع Persistence یا ماندگاری

نکته مهم این است که بعضی از منابع، منابع ماندگار هستند، مانند منابعی که بر روی دیسک قرار می‌گیرند. از جمله برنامه‌ها یا فرآیندهایی که خروجی آنها به دیسک منتقل می شوند و برخی غیرماندگار هستند، مانند منابع موجود در حافظه اصلی که با قطع شدن برق ممکن است از بین برود. در هر حال در Transparency از نوع

Persistence این مسأله که در واقع اینکه از کدامیک از این دو منبع استفاده می‌کنیم، باید از دید کاربر پنهان باشد.

Opening

با توجه به مباحث قبلی، در مبحث Opening باید سرویس هایی در اختیار دیگران قرار گیرد. حال به بررسی موارد مهم در Opening می پردازیم. اولین نکته در Opening بحث Protocol هاست. در واقع برای اینکه سیستم بتواند Open باشد باید Protocolهایی برای آن تدوین شود که این Protocol ها شامل قواعد استاندارد معانی پیغام ها و مفاهیمی مانند آن خواهد بود.

نکته ی دیگری که در این مبحث وجود دارد، مفهومی به نام IDL یا Interface definition language است، که به عنوان یک زبان تقریباً استاندارد تعریف واسط بکار می رود و معمولاً از این زبان برای ایجاد واسط‌های مورد نیاز برای دسترسی به سرویس‌های موجود در یک سیستم استفاده می شود. بنابراین یکی از ابزارهای مهم Opening استفاده از استانداردهایی مانندIDL است. نکته‌ی بعدی بحث عملیات متقابل(Inter operability) می باشد که بیان می کند، پیاده سازی‌های مختلف از سیستم‌های مختلف و اجزاء مختلف از سازندگان مختلف باید بتوانند همزمان با یکدیگر کار کنند و در سیستم حضور داشته باشند و مشکلی برای آنها به وجود نیاید. بنابراین می‌توانیم Inter operability را به عنوان کار بینابینی ترجمه کنیم که در یک سیستم توزیع شده به خصوص در نوع غیر همگن ممکن است که اجزاء مختلف از کمپانی های مختلفی باشد که با توجه به قواعد و استانداردهای تعریف شده به سادگی می توانند با یکدیگر Join شده و عملیات انجام دهند.

مبحث بعدی Portability یا قابلیت حمل است. به این معنی که باید سیستم به گونه ای طراحی شود که یک فرایند بدون هیچ تغییری بتواند بر روی دو سیستم عامل مختلف اجرا شود.

Flexibility یا قابلیت انعطاف در واقع مفهومی است که باید بتوانیم سیستم را با اجزاء مختلف از تولید کنندگان متفاوت پیکربندی کنیم. در واقع سیستم نباید وابسته به یک کارخانه خاص باشد. بنابراین اگر قواعد و

پروتکل‌ها رعایت شود و سیستم یک سیستم مناسب باشد آنگاه Flexibility، Interoperability ، portability به سادگی به دست خواهد آمد.

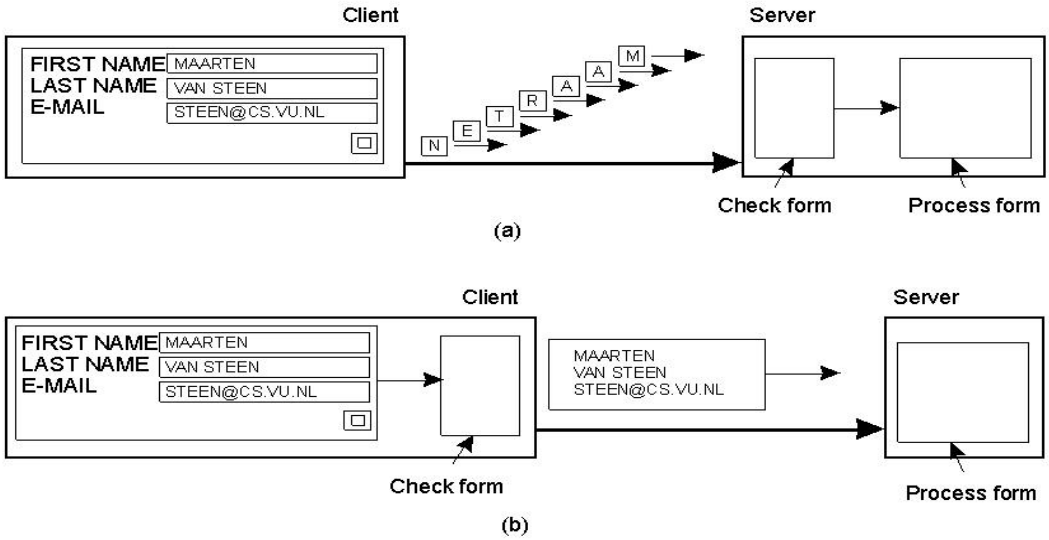
یکی دیگر از خاصیت‌هایی که در مورد سیستم های توزیع شده عنوان شد، مبحث Scalability یا قابلیت گسترش بود. Scalability از دیدگاه های مختلف قابل بررسی است از جمله این دیدگاه‌ها ، دیدگاه اندازه است. حال آنچه که این مسأله را برای ما جذاب می سازد این است که حتی اگر بتوانیم یک سیستم را به صورت متمرکز (Centralize) پیاده سازی کنیم، محدودیت های موجود در یک سیستم متمرکز به گونه ای خواهد بود که امکان گسترش سیستم را محدود خواهد کرد. یکی از مهم ترین مشکلاتی که در یک سیستم متمرکز رخ می‌دهد، محدودیت سرویس ها، داده‌ها و الگوریتم‌هائی است که در یک سیستم متمرکز وجود دارد. در واقع شما در یک سیستم می توانید حد مشخصی داده قرار دهید، سرویس های مشخص و معدودی را در اختیار دیگران قرار دهید و در واقع پس از مدتی امکان اضافه کردن کاربران جدید به یک سیستم وجود نخواهد داشت. از سوی دیگر برای رفع این مشکل استفاده از سیستم های توزیع شده پیشنهاد می شود بطوری که داده‌ها بر روی ماشین های مختلف قرار گیرد. اما نمی توانیم بگوئیم که با استفاده از سیستم های توزیع شده دیگر نیازی به سیستم های متمرکز نداریم، به این دلیل که سیستم های متمرکز به دلیل وجود امنیت بالا در آنها قابل چشم پوشی نیستند. بنابراین در یک سیستم، سرویس‌های متمرکز و نیز داده‌های متمرکز و حتی الگوریتم‌های متمرکز خواهیم داشت. به عنوان مثالی از سرویس های متمرکز حالتی را می توان در نظر گرفت که یک سرور به تنهایی به همه کاربران سرویس می‌دهد. بطوریکه بر روی آن سرور مجموعه‌ای از سرویس‌های متمرکز وجود دارد، و یا چنانچه یک کتاب تلفن آنلاین منفرد را در نظر بگیرید آنگاه مجموعه ی Centralize Data داریم و الگوریتم‌های متمرکز می توان به انجام مسیریابی بر اساس مجموعه ای از اطلاعات کامل اشاره کرد. نکته مهم در مورد الگوریتم‌ها و سیستم های متمرکز آن است که یک چنین سیستمی باید مجموعه ی زیادی از پیغامها را مدیریت کند و این مسأله، مسأله ی چندان خوشایندی نیست. بنابراین به سمت الگوریتم‌های نامتمرکز حرکت می کند. در الگوریتم های نامتمرکز نکات زیر را در نظر می‌گیریم. اول آنکه هیچ ماشینی اطلاعات کامل را درباره ی کل وضعیت سیستم ندارد. دوم آنکه ماشین ها تصمیم گیری های خود را بر

اساس اطلاعات محلی خود انجام می دهند. سوم آنکه خطای موجود در یک ماشین باعث متوقف شدن الگوریتم غیرمتمرکز یا توزیع شده نمی شود. چهارم آنکه در این الگوریتم ها وجود یک کلاک عمومی که همه‌ی سیستم ها در یک لحظه به آن دسترسی دارند معمولاً به عنوان یک فرض عمومی وجود ندارد. بدیهی است که این چهار خاصیت، خاصیت‌های مطلوب هستند و از سوی دیگر این نکته نیز بدیهی است که هر چه سیستم بزرگتر باشد عدم قطعیت موجود در سیستم نیز افزایش خواهد یافت. زمانیکه Scalability یا قابلیت بزرگ شدن را از دیدگاه جغرافیایی بررسی می‌کنیم، مفهومی تحت عنوان سنکرون کردن ارتباطات عیان می‌شود. معمولاً به عنوان یک مثال از این مفهوم نکته ای که باید به آن دقت کرد این است که اگر ما یک سیستم توزیع شده داشته باشیم ممکن است فرایندهایی در نقاط مختلف وجود داشته باشند که این فرایندها با یکدیگر رد و بدل داده و اطلاعات نمایند. چنانچه کسی از فرایند دیگری سرویس یا داده ای بخواهد آنها را به عنوان client شمرده و این client ها منتظر می‌مانند تا جواب را بگیرند و این انتظار یک انتظار پر مخاطره است. چه بسا که به دلیل مشکلاتی که در حوزه‌ی گسترده ی جغرافیایی یک شبکه جغرافیایی رخ داده است این انتظار ممکن است منجر به ایجاد بن بست و مفاهیم مشابه آن شود. از سوی دیگر در سیستم‌های Scalable جغرافیایی، ارتباطات نامطمئنی برقرار است، چرا که ممکن است پیغام هایی که معمولاً در این سیستم ها برای ارتباط با یکدیگر به کار می رود بر اثر مشکلات مربوط به ترافیک شبکه از بین برود. بنابراین نکته ی مهمی که وجود دارد این است که مفاهیم متمرکز در طراحی Scalability جغرافیایی نقش اساسی بازی می کنند.

تکنیک‌های قابلیت گسترش (Scalability)

اولین تکنیک، تکنیک‌هایی است که برای مخفی کردن اتلاف زمان ارتباطات مورد استفاده قرار می‌گیرد. ساده‌ترین تکنیک این است که انتظار برای پاسخ به یک سرویس از راه دور ممکن است کاهش یابد یا حذف شود. به عبارت دیگر چنانچه client درخواستی از فرآیند دیگری بر روی ماشین دارد که این درخواست را به عنوان یک درخواست راه دور می دانیم، آنگاه client بتواند پس از انجام درخواست، به کار خود ادامه دهد. در غیر این صورت این انتظار client از دیدگاه User به معنای این خواهد بود که User می تواند بفهمد که

client منتظر فرآیندی است که بر روی یک ماشین دیگر قرار دارد. از جمله راه هایی که برای چشم پوشی از این انتظار می توان ارائه داد استفاده از ارتباطات نا همگام است. این ارتباطات معمولاً در سیستم‌های Batch مناسب است. برخی از Application ها اصولاً نمی توانند Asynchronous باشند، بلکه برنامه‌های Interactive هستند. راه حل دیگر کاهش میزان ارتباطات است. به این صورت که یک راه، انتقال قسمتی از فرم محاسباتی سرور به client است. این مسأله باعث می شود که مجموعه ی اطلاعات ردوبدل شده بین سرور و client کاهش یابد. استفاده از سیستم های پایگاه داده نمونه ای از این حالت است.

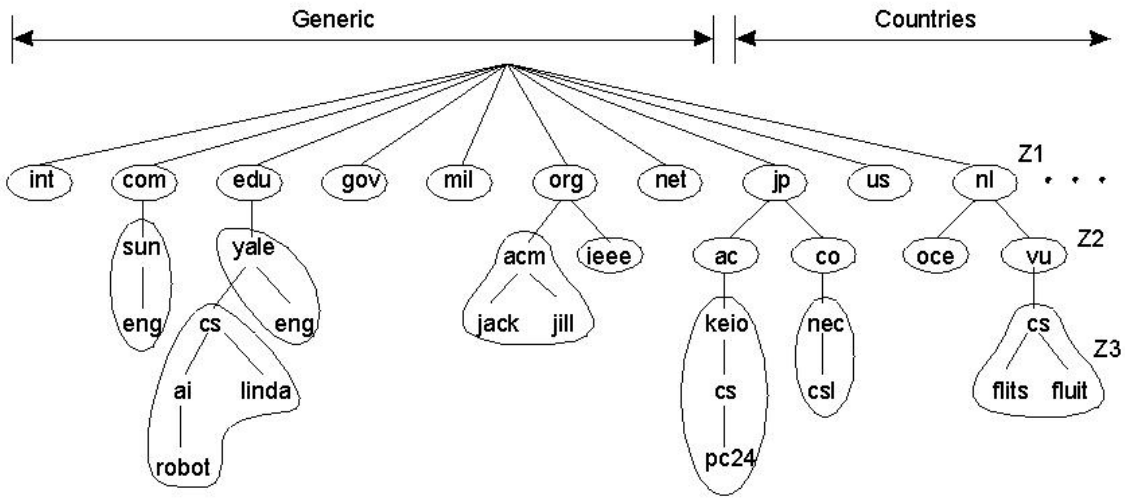


در شکل A سرور به عنوان کسی است که تمامی اطلاعات مربوط به یک فرم ارسال شده از سمت client را دریافت کرده و داده های آن را بررسی می کند. با توجه به آنچه در شکل می‌بینید، این اتفاق باعث می شود که سرور حجم زیادی از پردازش را انجام دهد. اما در شکل B با انتقال قسمتی از اطلاعاتی که در شبکه ی قبلی به سوی client منتقل می‌شد این مشکل حل شده است و در واقع بسیاری از پردازش هایی که سرور قبلاً انجام داده است در این زمان به بخش client منتقل شده است.

تکنیک های توسعه پذیری (Scalability Techniques)

از دیگر تکنیک های بهبود Scalability استفاده از مفهوم Distribution یا توزیع می باشد. هدف از مفهوم توزیع این است که داده ها یا سرویس ها را با توجه به نوع کاربرد آنها بر روی ماشین‌های مختلف توزیع نماییم.

به عنوان مثال ساختاری ارائه کنیم که برخی از ماشین ها فقط برخی از سرویس های خاص را ارائه دهند. به عنوان یک نمونه عملی تر که مفهوم توزیع را برای افزایش سطح توسعه پذیری یا Scalability یک سیستم توزیع شده مورد استفاده قرار می دهد مفهومی مشابه DNS ها یا (Domain Name Service) ها خواهد بود. همانطور که می‌دانید در DNS ها به ازاء هر وب سایت که کار خاصی انجام می دهد از Domain Name خاصی استفاده می شود. به عنوان مثال برخی از Domain ها، edu یا Education هستند که مربوط به مؤسسات آموزشی می‌باشند، gov مربوط به مؤسسات دولتی خواهد بود. حتی برخی از Domainها مربوط به کشورها هستند، به عنوان مثال دامنه ی iri مربوط به ایران، u.s مربوط به آمریکا و.....، لذا همانطور که در شکل نشان داده شده است DNS می‌تواند یک مفهوم توزیع شده را نشان دهد که سرویس ها بر حسب مورد آنها در قسمت‌های مختلف قرار گرفته است و به عنوان یک درخت سرویس دیده شود.



بنابراین استفاده از توزیع داده ها و سرویس ها به بحث توسعه و امکان توسعه دادن یک سیستم توزیع شده کمک خواهد کرد.

تکنیک بعدی که برای Scalability بکار برده می‌شود مفهوم Replication یا تکرار است. منظور از آن ایجاد نسخه‌های متعدد از یک سرویس و یا یک داده در قسمت‌های مختلف شبکه است. به عنوان مثال یک بخش داده‌ای در چندین سرور وجود خواهد داشت. استفاده از Replication به شدت مورد توجه است و باعث افزایش کارایی خواهد شد. در استفاده روزمره از شبکه ها معمولاً Replication را به کار می برند. به عنوان مثال، می توان به سرویس های ارائه شده توسط سرورهای بزرگ اینترنتی مانند yahoo یا gmail که از تکنیک Replication استفاده می کنند اشاره کرد. در واقع با استفاده ازچنین تکنیک هایی، ممکن است شما در هر لحظه در هر مکانی که قرار دارید به سرورهای متعددی که محتوای آنها مشابه یکدیگر است متصل شوید. نکته بسیار حائز اهمیت در بحث Replication مفهوم Consistency است. بدین معنا که نسخه های متعددی که از یک داده وجود دارند، بطور حتم با استفاده از تکنیک‌های خاصی با یکدیگر سازگار شوند. به این معنی که چنانچه تغییری در یکی از این نسخه ها داده شود نسخه های دیگر نیز سریعاً بتوانند به روز شده و حداقل این تغییر را در خود داشته باشند.

از جمله مزایای دیگر استفاده از Replication ، مقاوم سازی سیستم در مقابل خطاست. این مفهوم بدان معناست که چنانچه به هر دلیلی یکی از نسخه های مربوط به داده ها از بین برود سیستم Done نخواهد شد و کماکان امکان ارائه سرویس با سطح پائین تری از Performance به کاربران وجود خواهد داشت. یکی از مثال‌های ملموس‌تر Replication استفاده از مفهوم Caching است. Caching در واقع به نوعی Replication است. به این معنی که داده هایی که در حافظه قرار دارند در حافظه ی دیگری که سرعت دسترسی به آن بالاتر است کپی می شوند. ارائه سرویس ها از داخل حافظه ی Cache صورت می گیرد.