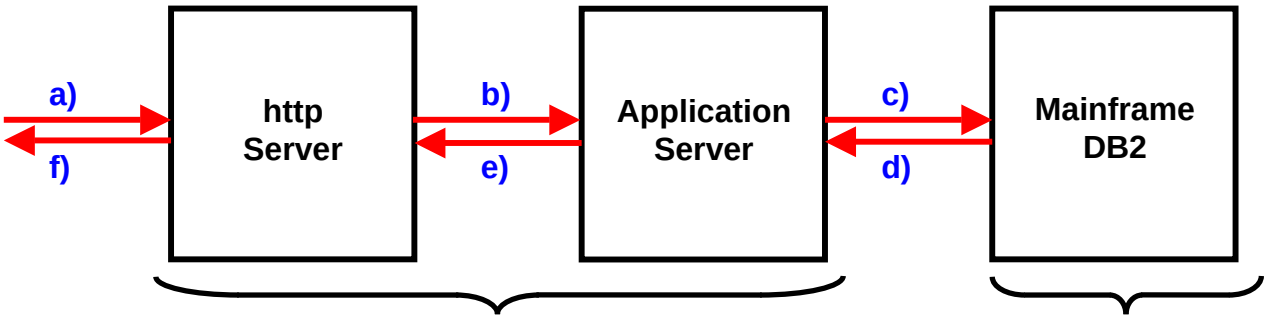
Besprechungsnotiz zum Thema WLM Projekt: 19.12.2011
Teilnehmer: Manfred Bogdan (Telefonkonferenz), Peter Bäuerle, Wolfgang Dunz, Horst Sinram, Wilhelm Spruth, Stefan Wirag .

Problemstellung

Herr Simram und Herr Wirag stellten die Problemstellung am Beispiel der skizzierte Client/Server Konfiguration vor. Angenommen ist ein HTTP Server und ein Application Server. Letzterer kann z.B. Middleware wie WebSphere sowie Servlets, JSPs und EJBs und JTA (Java Transaction API) enthalten. Die EJBs benutzen z.B. z/OS DB2, und evtl. weitere z/OS Middleware Funktionen. Alternative läuft auf dem Application Server Middleware wie Tuxedo, distributed CICS, SAP R/3, DCE oder Message Based Queuing (z.B. MQSeries).



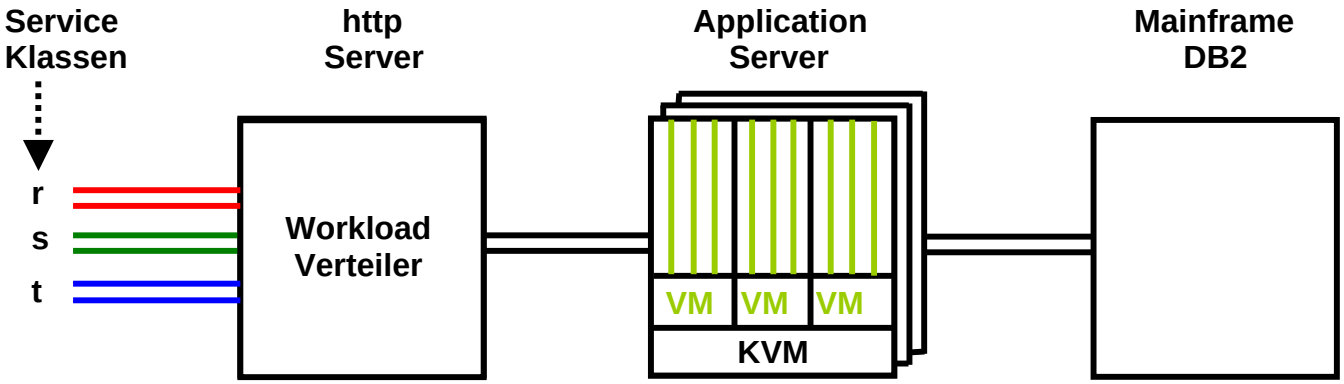
Der z/OS WLM kann die Response Time für eine gegebene Service Klasse von Punkt c) bis zu Punkt d) steuern. Eine Steuerung ist nicht machbar von Punkt a) bis bis zu Punkt f). Spezifisch ist auch nicht zu ermitteln, wie sich die beobachtete gesamten Response Time von a) bis f) zwischen Application Server Response Time und Mainframe Response Time aufteilt.

Der http Server enthält typischerweise eine Workload Verteilungsfunktion (im einfachsten Fall einen Sprayer), welche die eintreffenden Transaktionen auf einen oder mehrere Application ServerServer aufteilt.

Wünschenswert ist die Einordnung aller eintreffenden Transaktionen in mehrere Service Klassen (hier als r, s und t bezeichnet) mit unterschiedlichen Vorgaben für die Wichtigkeit (Importance) sowie die Response Time (Delay) zwischen den Messpunkten a) und f).

Hierzu ist eine Instrumentierung für den Application Server und den http Server erforderlich. Messtechnisch zu erfassen sind spezifisch die Hauptspeichernutzung, die Fehlseitenrate (paging), die CPU Nutzung sowie die Ein/Ausgabe (I/O) Rate für die angeschlossenen Plattenspeicher. Die Nutzung dieser Betriebsmittel muss für jede Service Klasse getrennt erfasst werden.

Falls auf dem Application Server ein Betriebssystem wie Windows, Solaris oder HP-UX läuft, ist es nicht möglich, den erforderlichen Instrumentierungscode in den Betriebssystem-Kernel einzubauen. Dies gilt ebenso für die eingesetzte Middleware.



Zur Lösung wird vorgeschlagen, alle Anwendungen des Application Servers in virtuellen Maschinen (VM) unter einem Hypervisor, z.B. KVM, laufen zu lassen. Da KVM Open Source ist, können Instrumentierungsfunktionen eingebaut werden. Dem Hypervisor sind Informationen wie CPU Nutzung, Hauptspeichernutzung, Fehlseitenraten und I/O Anforderungen für jede virtuelle Maschine zugänglich. Dabei ist es unerheblich, welches Betriebssystem und welche Middleware unter dem Hypervisor läuft.

Hierzu werden alle Transaktionen in Service Klassen r, s, t mit unterschiedlichen Goals (response time, importance) eingeordnet. Alle eintreffenden Transaktionen werden vom Workload Verteiler mit einem entsprechenden Tag versehen.

Unter KVM können auf jedem physischen Rechner viele virtuelle Maschinen laufen. Es werden auch sehr häufig mehrere physische Server eingesetzt. Eine mögliche Lösung (hier als Ansatz A bezeichnet) besteht darin, für jede Service Klasse eine eigene virtuelle Maschine einzurichten, wobei auf jeder virtuellen Maschine zahlreiche Prozesse und/oder Threads für die parallele Verarbeitung der Transaktionen dieser Serviceklasse laufen. In diesem Fall ist es möglich, Daten über die Hauptspeichernutzung, die Fehlseitenrate (paging), die CPU Nutzung sowie die Ein/Ausgabe (I/O) Rate jeder einzelnen virtuellen Maschine (und damit jeder Service Klasse) zu ermitteln, und an den z/OS WLM zwecks weiterer Verarbeitung zu übergeben. Das gilt auch, wenn mehrere virtuelle Maschinen für die gleiche Service Klasse zur Verfügung stehen, z.B. auf getrennten physischen Servern.

Derzeitig vermuten wir, dass Ansatz A in vielen Fällen nicht gangbar sein wird, weil die Anzahl der Service Klassen (typischerweise < 25 – 30) die Anzahl der virtuellen Maschinen übersteigt, oder die starre Zuordnung von Service Klassen zu virtuellen Maschinen nicht möglich ist. In diesem Fall muss davon ausgegangen werden, dass die parallel ablaufenden Prozesse und/oder Threads einer bestimmten JVM in jedem Augenblick Transaktionen unterschiedlicher Service Klassen verarbeiten. Da der Hypervisor aber nur Resource-Verbrauchs-Informationen für die Prozesse oder JVMs insgesamt, nicht aber für die einzelnen Prozesse/Threads oder JVMs erhält, ist eine Ressourcen Nutzung für die einzelnen Service Klassen nicht zu ermitteln.

In dem hier diskutierten Ansatz B enthält der Hyperviser eine Mustererkennungskomponente, welche Rückschlüsse auf die Ressourcennutzung der einzelnen Service Klassen ermöglichen soll. Welche Daten als Input für die Mustererkennungskomponente eingesetzt werden, muss im Rahmen des Projektes noch erforscht werden. Dies sind einige Möglichkeiten, die mir einfallen:

- **Veränderung der Anzahl der Transaktionen (pattern), und beobachtete Änderungen in der Response Time (Durchschnitt, Standard Deviation), für eine bestimmte Service Klasse von einem Intervall zum nächsten.**

- **Änderung der prozentalen Verteilung der Transaktionen (Mix) auf die einzelnen Service Klassen und beobachtete Änderungen in der Response Time.**

- **Häufigkeit von Zugriffen auf einzelne Plattenspeicherbereiche und versuchte Zuordnung zu Serviceklassen.**

- **Untersuchung von Vanderpool/Pacifica Daten (z.B. wie häufig schaltet eine virtuelle Maschine vom Host- in den Gaststatus), sowie versuchte Zuordnung zu Serviceklassen.**

- **Beobachtung der virtuellen nach realen Adressumsetzung. Auslesen des TLB (Table Look-Aside Buffers) durch den Hypervisor. Ich nehme an, dass dies heute aus Diagnostik Gründen möglich ist.**

- **Unter allen Unix Gast Betriebssystemen wäre es denkbar, einen zusätzlichen Monitor Prozess zu installieren, der im User Status ein Script mit dem „ps“ Unix Kommando enthält und damit auf /proc/pid# der virtuellen Maschine zugreift. Hier ist Prozess-spezifische Information vorhanden, die an die Hypervisor Mustererkennungs-komponente übergeben werden könnte. Das ps Kommando ist auch unter Linux verfügbar. Ich kann mir vorstellen (tbd), eine mit dem ps Unix Kommando vergleichbare Einrichtung ist auch unter Windows vorhanden.**

- **????**

Die Mustererkennungskomponente könnte damit versuchen, die Wahrscheinlichkeit für die Nutzung von Ressourcen durch die Transaktionen einer bestimmten Service Klasse zu ermitteln. Neuronale Netze bieten sich hierfür an. Dies wird dann in geeigneter Form als Input für den z/OS WLM benutzt.

Unklar bleibt diese Frage: Wir könnten die verschiedenen WebSphere DataPower Appliances relativ leicht instrumentieren, da wir Owner des Codes sind. Wäre das nicht eine für den Vertrieb wichtige Differenzierung gegenüber Mitbewerber Produkten mit ähnlicher Zielsetzung ?

Anderes Beispiel: Wir könnten IBM x-Blades mit zusätzlichen Hardware Hooks versehen, welche die Instrumentierung erleichtern würden. Auch das würde ein IBM Alleinstellungsmerkmal sein.

What next ?

Ich schlage ein Treffen der Herren Bogdan, Bäuerle, Dunz und Spruth in der 1. Januarwoche in Tübingen vor. Bitte Terminvorschläge, ich selbst bin sehr flexibel.

Agenda:

1. **Gedankenaustauch**
2. **Brainstorming**
3. **Diskussion und Beschluss über weiteres Vorgehen.**

Unter anderen möchte ich die Erstellung eines Ansatz A Prototypen mit konkreter Anbindung an unseren Mainframe zur Diskussion stellen. Dies könnte auf einem Linux Rechner des Lehrstuhls Technische Informatik erfolgen, den wir an unsere z9 anbinden würden, und der remote programmiert und bedient werden kann. Das würde ermöglichen:

- **Austesten**
- **Verständnis verbessern**
- **Erfahrung sammeln**

Dieser Prototyp würde Test Daten vom Labor erfordern. Zumindest die Planung eines derartigen Prototypen würde helfen, das Verständnis zu vertiefen und offene Fragen an die Oberfläche zu bringen. Weiterhin könnte der Ansatz A Prototyp als Testbed für einen späteren Ansatz B Prototyp dienen.

21. 12. 2011, W.G. Spruth