Heterogeneous Workload Management

IBM

Donna Dillenberger
What is Heterogeneous Workload Management?

- Heterogeneous Workload Management (HWLM) is system management software that will manage different kinds of servers
- trying to bring all computing resources together online, discover applications and hardware configuration automatically and use all resources more efficiently.
How does this fit in the autonomic space?

- It uses "self-learning" software that enables servers to manage themselves without human intervention.
- As companies use hundreds to tens of thousands of servers, the ways to configure all of them is astronomical.
What are the biggest challenges?

• Creating adaptive algorithm that can prioritize learned knowledge and scale all sort of problems is one of our challenges
What are the differences between HWLM and Océano?

• While Océano handles Web application servers, HWLM addresses all kinds of servers across the network, including edge, appliance, transaction and database servers.

• Océano will become part of HWLM to do its specific tasks: primarily adding or removing servers as the workload dictates.
What is the timeline for this work? How soon can we see HWLM in the real world?

- The software is working very well, managing more than 800-1,000 servers automatically.
- They expect the first products incorporating HWLM technology in 2002.
Adaptive algorithms

IBM

J. Aman, C. K. Eilert, D. Emmes, P. Yocom, D. Dillenberger
Workload management

• a function of the OS/390* operating system base control program, allows installations to define business objectives for a clustered environment (Parallel Sysplex* in OS/390)

• This paper presents algorithms developed to simplify performance management, dynamically adjust computing resources, and balance work across parallel systems
Related Work

- Resource management
- Workload balancing
Resource management

- No controls
- “owner” && “foreign”
- Minimize response time
Workload balancing

- WLM tracks those factors needed to best place incoming work and provides interfaces to make workload-balancing recommendations
WLM system model

- A service class is comprised of a sequence of periods
- Each work request starts in period 1 and is managed according to the first period goal
- Each period has an associated goal and an associated importance
goal types

- response time
- discretionary
- velocity
Response time (goal)

- indicate a desire for internal elapsed time to be
- elapsed time refers to wall-clock time
Discretionary (goal)

• indicates that there is no business requirement for the work to complete within a certain predetermined elapsed time
Velocity (goal)

• Work requests that are not considered discretionary and do not have a set response time objective nevertheless may need further control to reflect the degree of delay that is tolerable
importance

• Importance is merely a relative ranking of work and is only a factor
• whose goals will be attended to first when system resources are reallocated
<table>
<thead>
<tr>
<th>Service Class Period</th>
<th>Type of Goal</th>
<th>Goal</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICSTRX</td>
<td>Response time</td>
<td>0.090 sec</td>
<td>Medium</td>
</tr>
<tr>
<td>IMSTRX</td>
<td>Response time</td>
<td>1.000 sec</td>
<td>High</td>
</tr>
<tr>
<td>TSO Period 1</td>
<td>Response time</td>
<td>0.100 sec</td>
<td>Medium</td>
</tr>
<tr>
<td>TSO Period 2</td>
<td>Response time</td>
<td>1.000 sec</td>
<td>Medium</td>
</tr>
<tr>
<td>TSO Period 3</td>
<td>Response time</td>
<td>3.000 sec</td>
<td>Low</td>
</tr>
<tr>
<td>BatchHi</td>
<td>Velocity</td>
<td>7%</td>
<td>Lowest</td>
</tr>
<tr>
<td>BatchLow</td>
<td>Velocity</td>
<td>1%</td>
<td>Lowest</td>
</tr>
</tbody>
</table>
WLM algorithms for resource management

- MGDPC
- Fundamental concepts
- Policy adjustment framework
Multisystem Goal-Driven Performance Controller (MGDPC) 1

- a data collection and analysis system, extending across a set of interconnected, cooperating, independent computer systems
- invoked once every ten seconds
Multisystem Goal-Driven Performance Controller (MGDPC) 2

- collects performance data
- measures the achievement of goals
- selects the service classes that need their performance improved
- selects bottleneck resources
- selects donors of the resources
- assesses the impact of making resource reallocations
- makes the reallocations if there is a net benefit to the changes
Fundamental concepts

- Data histories -- a mechanism to collect and analyze data over time
- performance_index = \frac{actual\_response\_time}{goal\_response\_time}
- State sampling
- Server topology
Figure 1  State sample types
Figure 2  Client/server diagram

CLIENT SERVICE CLASSES

CICSA

IMPORTANCE 3
RT GOAL = 2 SEC

INTERNAL SERVICE CLASSES

INTERNAL CLASS 1

AOR1
SERVES CICSA

INTERNAL CLASS 2

TOR1  AOR2
SERVES CICSA AND CICSB

INTERNAL CLASS 3

AOR3
SERVES CICSB
Policy adjustment framework

- The policy adjustment algorithm is invoked periodically to assess reallocating system resources to better meet performance goals
Figure 4  Policy adjustment algorithm loop

SELECT A RECEIVER

FIND A RECEIVER'S BOTTLENECK

FIX RECEIVER'S BOTTLENECK
A. SELECT DONORS OF REQUIRED RESOURCE
B. ASSESS EFFECT OF REALLOCATING RESOURCES
C. IF REALLOCATION HAS VALUE, COMMIT CHANGE

IF REALLOCATION WAS DONE

IF MORE BOTTLENECKS

IF MORE RECEIVERS

EXIT
Assemble performance data

- At the beginning of each policy interval, performance data that have been collected asynchronously by state sampling
- prepare for running the adjustment algorithms
Select receiver

• The first decision the policy adjustment algorithm must make is to decide which class to help
Find bottleneck

• Once the receiver class has been selected, the next step is to select which resource delays to address
Generic delay fix

- There is a specific fix algorithm for each delay addressed by the MGDPC. The function of each fix algorithm is to improve the performance of the receiver class
Select donor

- The purpose of the select donor algorithm is to choose the most eligible class that will donate the required resource to the receiver from the set of classes owning that resource
Net value

• The net value algorithm keeps the MGDPC from making bad resource reallocations
Send data

• At the end of each policy interval on each system, the MGDPC sends data to all the other systems in the set of independent cooperating systems being managed
Benefit 1

- the business policy defined to WLM handles mixed workloads
- e.g., interactive, batch, transaction processing, data mining environments, and so forth
Benefit 2

• The second major benefit is to support high *availability* objectives
• This support includes rebalancing work when an image is removed and advising in the placement of restarting subsystem environments when their host system is removed
Benefit 3

- WLM philosophy is to require no changes at the application level
Cited references and notes