Global Resource Director (GRD)

Customer Scenarios for Large Multiprocessor Environments

by

Fritz Ferstl

GENIAS Software GmbH

Abstract

Genias Global Resource Director (GRD) was developed to meet the demands of large multiprocessor sites with several 100s of CPUs and users. At the Army Research Laboratory (ARL) and at BMW, GRD is installed on several types of Cray and SGI machines. This paper will describe how GRD integrates such resources into one environment automatically managed by global resource utilization policies. Benefits which the sites achieved in using GRD such as avoiding over-subscription of resources, automated management and utilization monitoring will also be discussed. In addition, an overview will be provided on the ground-breaking functionality of GRD which features a ticketing system based distribution of resource shares combined with on-line control of the CPU utilization of running jobs (also known as “dynamic scheduling”).

1 GRD Overview

GRD originated from a joint development of Raytheon Systems Company (formerly E-Systems) of Garland TX, Instrumental Inc. of St. Paul MN, and GENIAS Software GmbH of Neutraubling Germany. The product is unique in its approach to provide more effective management of distributed computer systems. This article describes the product’s origins, capabilities, benefits, and applications.

GRD is intended for computing centers in need of more effective resource management and policy administration and, in particular, heterogeneous or homogeneous UNIX environments that involve multiple major shared resources. GRD leverages complementary technologies of three companies to bring about major improvements over existing products including:

- Increased automation and control over resource usage and associated policy administration
- Higher resource utilization and throughput (and associated ROI)
- Reduced administrative and operational workloads
- Improved level-of-service and higher productivity for users

These improvements result from the unification of three primary capabilities:

(1) CODINE, a proven *best-in-class* job management system from GENIAS Software GmbH

(2) Innovative dynamic scheduling and resource management software from Raytheon E-Systems known as Global Dynamic Scheduler (GDS)

(3) Dynamic performance data collection software from Instrumental Inc.

These technologies are tightly integrated to provide a unique solution to the complex resource management needs of shared resource centers while offering many advanced features including:

- **Multiple workload management policies**--GRD supports fair share, proportional share, priority, and deadline based policies. Extensive flexibility is provided for customizing the use/mix of these different policies.
- **Automated policy enforcement**--GRD automates the use of multiple policies (e.g., share based, deadline) and multiple importance levels (e.g., routine, urgent, high priority). Power users and ad hoc urgent requirements are accommodated without operational disruption or manual reconfiguration of workloads. GRD automatically mediates all resource usage to address each processing need.
- **Global resource management**--The various policies govern the allocation of a center's total resource capacity among competing jobs, processes, users, projects, departments, and job classes in addition to the allocations within each host. By maintaining a global view of resources and users, GRD can prevent (or allow) individuals from dominating resources in the short term (i.e., at any given moment) and the long term (i.e., over a week, month, etc.) while ensuring that high importance, time-sensitive, and ad hoc processing needs are met.
- **Proportional share resource management**-GRD implements a flexible ticket-based proportional share resource management system that integrates multiple policies in a single resource management engine [KAY88, WALD94]. The resource management engine continuously monitors resource usage and exercises low level priority control over the and exercises low level priority control over the entire workload to mediate resource usage among competing jobs and maintain alignment with policies.
- **Share-based global resource management and job scheduling**--GRD maintains resource usage statistics for users or projects and is able to automatically keep long term resource utilization in line with a site-specific hierarchical resource allocation plan.
- **Function-based global resource management and job scheduling**--GRD supports policies for regulating resource utilization based upon the relative entitlements of users, projects, departments, and job classes.
- **Urgency-based resource management and job scheduling**--Run-time deadlines are integrated into the proportional share resource management system. This allows high urgency work to dominate resources only when necessary. Deadline-oriented workloads can be submitted early for background execution to take advantage of underutilized resources. GRD will escalate the proportional resource shares these jobs receive over time.
- **Enhanced load balancing**--GRD offers more effective load balancing and job migration by considering not only current resource loading factors and needs, but relative importance and urgency of jobs, and the resource entitlements of other jobs competing for the same resources.
- **Advanced job queuing environment**-- GRD is based on the CODINE job management system [COD197, COD297] and inherits the rich feature set of CODINE. CODINE is the defacto standard job management environment in Europe [CASA96] and has been rated as a first tier product in third party evaluations in the U.S. [KAPL94, BAKE95, JON97].
- **Centralized control**--Multiple policies are administered and utilized through a cohesive system of GUIs. All administration and operational changes can be accomplished through centralized controls without requiring the user to address individual host environments. An override system allows any of the policy parameters to be adjusted for global application and/or to allow allocated shares to be proportionally adjusted among multiple executing jobs for the same user, project, department, or job class. GRD automatically flows overrides and changes to the applicable workloads on the applicable hosts.

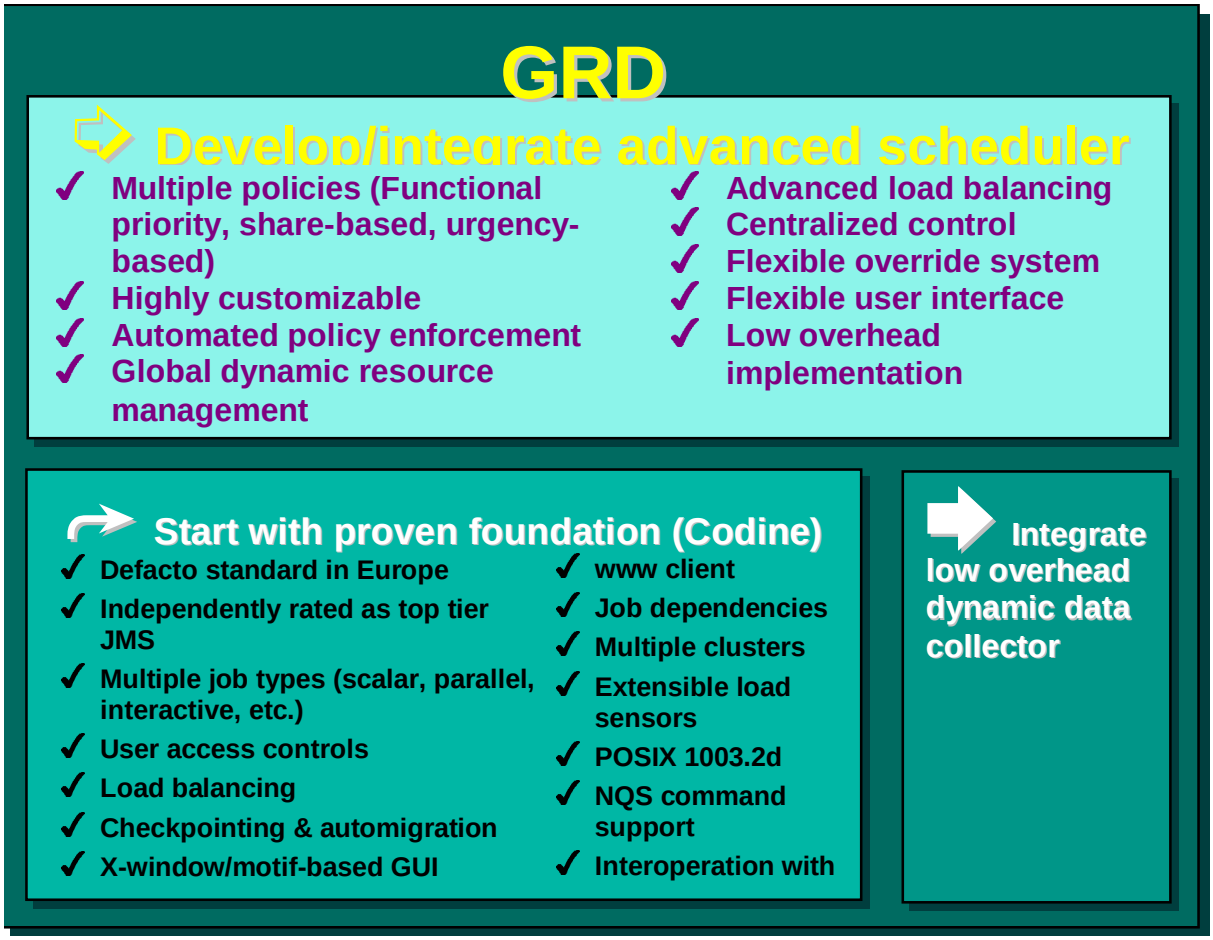

Figure 1: GRD Extends the CODINE Job Management System

The material that follows discusses the issues that have heretofore limited the effectiveness of workload management systems along with details of the GRD approach.

1.1 Problem Description

Workload management is the process of controlling the use of shared computer resources to maximally achieve the performance goals of the enterprise (e.g., productivity, timeliness, level-of-service). This is accomplished through resource management polices and tools that strive to maximize utilization and throughput while being responsive to varying levels of timeliness and importance.

To accomplish these goals, management solutions must utilize the relative importance of competing jobs and correlate concurrent instances of the same users, jobs, projects, etc. in order to implement effective resource sharing policies. Systems that lack this sophistication will have inherent weaknesses in mediating the sharing of resources such as:

- Applications will rarely perform at the optimum performance because imbalanced load is the common situation in multiprocessing environments, not the exception.
- Important/urgent work may be deferred or starved for resources while other work is initiated and processed.

- Unauthorized users may inadvertently dominate shared resources (and decrease productivity) by simply submitting the largest amount of work.
- A user may grossly exceed her/his desired resource utilization level over time.

These limitations lower overall resource utilization, reduce throughput performance (especially when relative priority is considered), and increase the need for operational or administrative intervention.

Furthermore, initial job placement occurs at a single point in time and cannot take natural and unforeseen dynamics into account. Actual resource usage can diverge significantly from desired usage after dispatching. Therefore, dynamic re-allocation of resources is a prerequisite to optimal workload management.

1.2 Solution

To avoid improper dispatching of jobs, GRD performs resource tasking based upon the utilization and collective capabilities of an entire system of resources and with complete awareness of the total workload. Jobs executing anywhere in the pool of managed resources are correlated with users, projects, departments and job classes in determining how to allocate resources when new jobs are submitted or complete.

The following table displays the drawbacks of current queuing system implementations with respect to resource tasking (the dispatching decision). The table shows how these issues are resolved in GRD and which benefits are obtained by using GRD.

Current Implementations	Resulting Limitation	Solution	Results	Benefits
• No correlation among multiple instances of same user, project, department, job class, etc. • Spawned processes not counted in workload associated with job • Current resource loading is primary parameter for job placement	• Users inadvertently dominate resources • Important work does not get appropriate share of resources • Poor job placement/load balancing • High importance work not properly distributed • Proper precedence not implemented in dispatching sequence	• Global view of resources and workloads with all workload elements properly correlated • Proportional dynamic scheduler • Multiple resource management policies • Dispatching decisions consider global resource shares, resource loading, intended usage, and resource capabilities • Dispatching precedence based on resource entitlement • Global redistribution of tickets after dispatching and/or job completion	• Utilization by specific users, projects, departments, etc. stays aligned with policy • More optimum fit of workload priorities to resources • More urgent work handled accordingly • Reduces need for manual intervention • Reduces need for job migration	• Higher effective utilization of resources • System more responsive to relative importance of jobs and workloads • Improves/automated control of long term usage • Users more satisfied with level of service • Reduces administrative and operational workloads

Table 1: Solving Current Resource Tasking Issues

But GRD does not only take resource utilization policies into account when dispatching jobs. GRD continuously maintains alignment of resource utilization with policies using a dynamic workload regulation scheme. GRD monitors and adjust resource usage correlated to all processes of a job, the corresponding job classes, users, projects, and departments. Operational adjustments to resource entitlements are centralized and take effect globally because of workload correlation and workload regulation.

The following table shows again drawbacks of current static scheduling tools as opposed to GRD´s dynamic workload control..

Current Implementations	Resulting Limitation	Solution	Results	Benefits
• Little or no automated control of workload execution after dispatching • Use of priority without workload regulation • Workload elements not correlated across resource pool	• Actual resource utilization diverges from intended utilization • Unreliable relationship between control parameters and workload performance • Excessive manual intervention • Unnecessary deferred execution of demanding jobs • Resources under-utilized	• Continuous monitoring of resource utilization • Global view of resources and workloads with all workload elements properly correlated • Ticket-based scheduler and proportional dynamic resource manager able to redistribute work shares • Utilize low level controls for controlling workload performance • Centralized override control	• Utilization by specific users, projects, departments, etc. stays aligned with policy • More urgent or more important work given appropriate share of resources • Reduces need for job migration • Reduced need for manual intervention • Demanding low priority jobs can be submitted early to take advantage of unused capacity	• Higher effective utilization of resources • System more responsive to relative importance of jobs • Improved/automated control of long term usage • Users more satisfied with level of service • Reduces administrative and operational workloads • More effective controls for administrative and operational intervention

Table 2: Solving Current Workload Regulation Issues

Figures 2 and 3 depict in further detail the difference between **static scheduling** as performed by current state-of-the-art resource management systems and the **dynamic scheduling** scheme implemented by GRD in order to solve the problems described above.

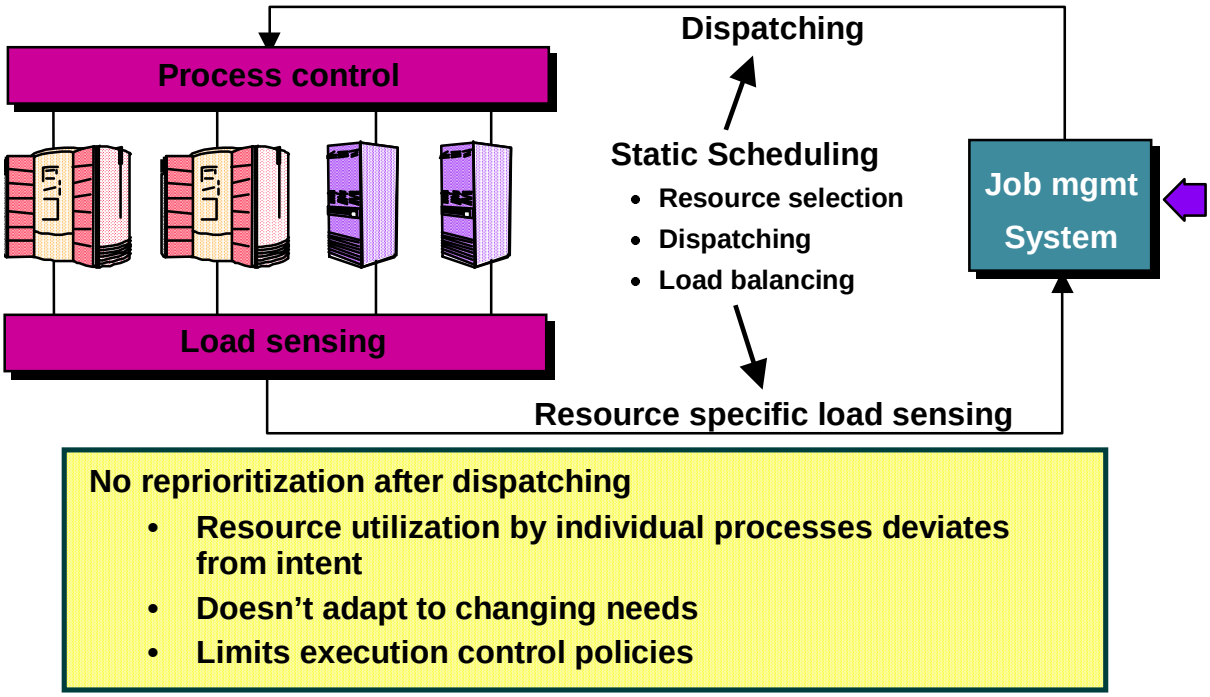

Figure 2: Workflow of current tools with static scheduling

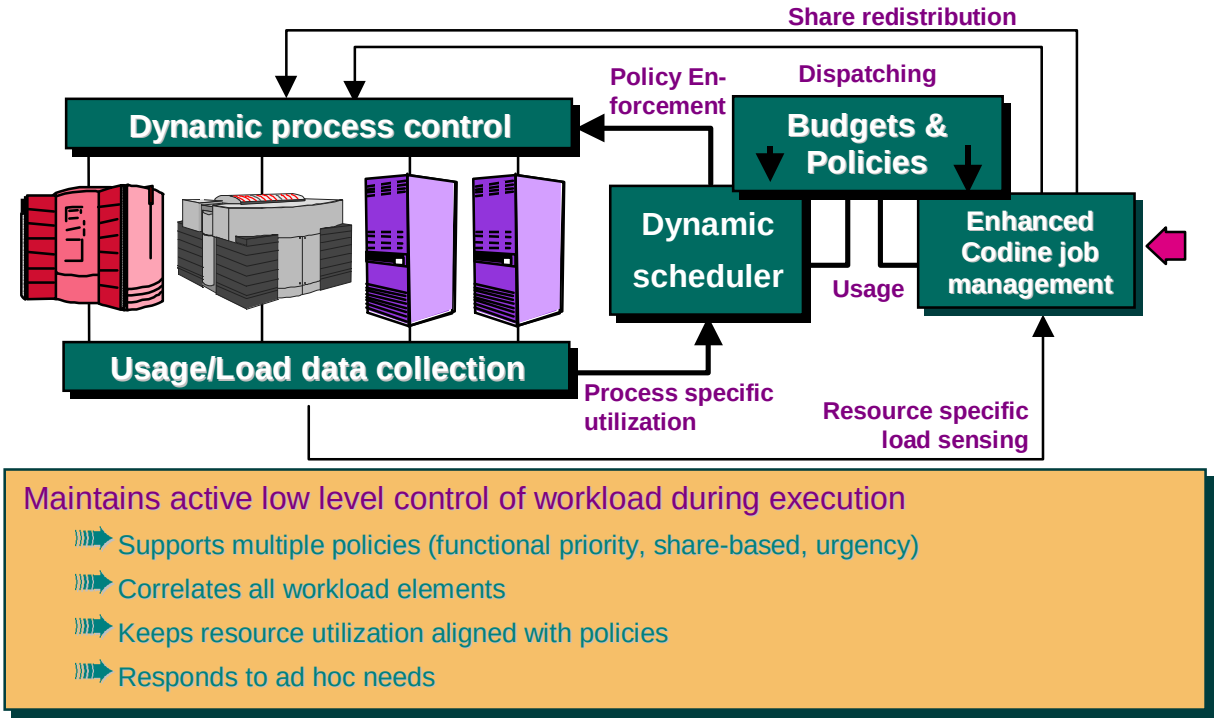

Figure 3: GRD workflow with dynamic scheduling feed-back loop

In order to be capable of adapting to a large variety of site specific resource utilization requirement GRD will schedule jobs using a weighted combination of policies:

- **Share-based**--Supports hierarchical allocation of resources to users or projects during a configurable time period that typically exceeds the life time of individual jobs
- **Functional**--Supports relative weighting among users, projects, departments, and job classes during execution
- **Initiation deadline**--Automatically escalates a job's resource entitlement over time as its deadline nears
- **Override**--Adjusts resource entitlements at the job, job class, user, project, or department levels
- **User discretionary**--Allows users to adjust relative importance of jobs within a user's workload

These policies can be combined as depicted in figure 4 using GRD´s **ticketing system**. Each policy is assigned a total amount of **tickets** which defines the relative level of importance of the policies. Tickets can be compared with stock shares – the more shares a stock owner has, the more influence s/he has.

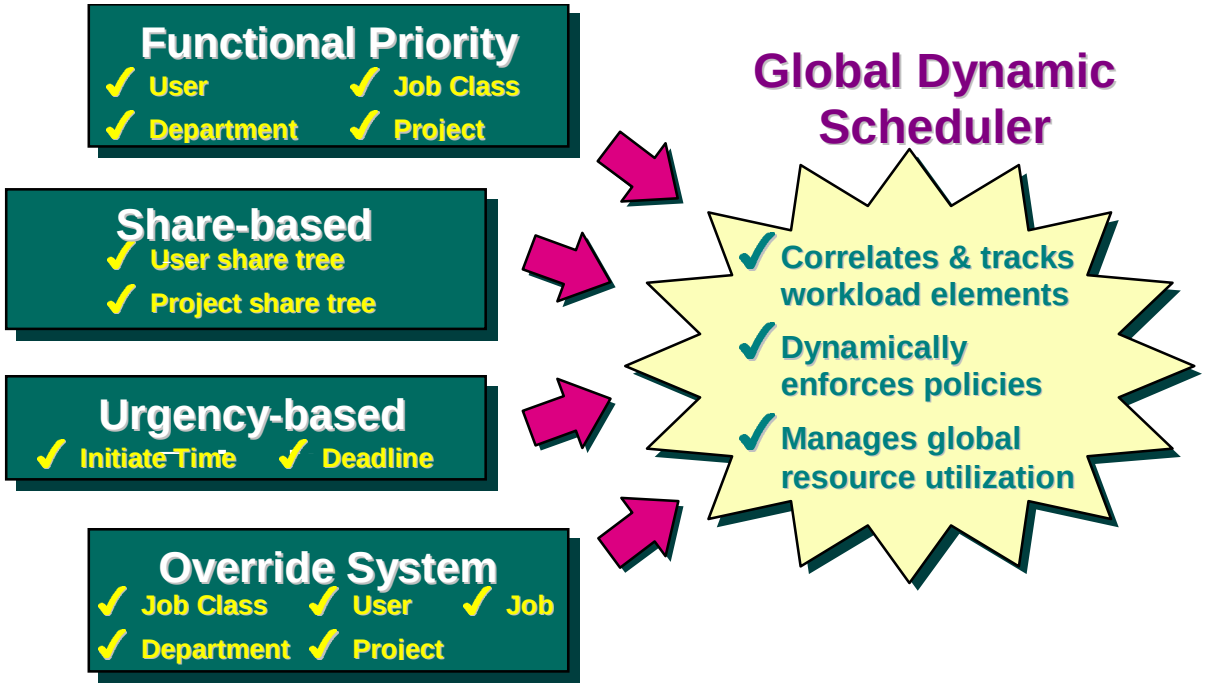

Figure 4: GRD Resource Utilization Policies

While the relation between the different policies is defined at the top level, the policies internally collect resource usage for users, projects, departments and job classes and GRD arbitrates the usage against the policies as defined by the site management. Thus when a single user (or project or department) will submit many jobs, other jobs that the user (or project or department) has in execution will receive fewer share allocation in order to keep associated resource usage at the appropriate level. Conversely when a job completes, the resource entitlements of other executing jobs for the associated user are escalated.

In determining **dispatching precedence** GRD will favor jobs with larger resource entitlements reflecting the fact that larger entitlements indicate higher importance.

1.3 Resulting Benefits

- **Multiple workload management policies**—GRD supports fair share, proportional share, priority, and deadline based policies. Extensive flexibility is provided for customizing the use/mix of these different policies.

- **Automated policy enforcement**—GRD automates the use of multiple policies (e.g., share based, deadline) and multiple importance levels (e.g., routine, urgent, high priority).

- **Proportional share resource management**--The resource management engine continuously monitors resource usage and exercises low level priority control over the entire workload to mediate resource usage among competing jobs and maintain alignment with policies.

- **Share-based global resource management and job scheduling**—GRD maintains resource usage statistics for users or projects and is able to automatically keep long term resource utilization in line with a site-specific hierarchical resource allocation plan.

- **Function-based global resource management and job scheduling**—GRD supports policies for regulating resource utilization based upon the relative entitlements of users, projects, departments, and job classes.

- **Urgency-based resource management and job scheduling**--Run-time deadlines are integrated into the proportional share resource management system. This allows high urgency work to dominate resources only when necessary. Deadline-oriented workloads can be submitted early for background execution to take advantage of under-utilized resources. GRD will escalate the proportional resource shares these jobs receive over time.

- **Centralized control**--Multiple policies are administered and utilized through a cohesive system of GUIs. All administration and operational changes can be accomplished through centralized controls without requiring the user to address individual host environments.

2 GRD at BMW

BMW has been using CODINE and later GRD since 1994 to distribute computing workload for crash simulations. GRD is today installed on 11 SGI Origin server systems and more than 100 SGI workstations with altogether 370 CPUs. The following sections sum up the benefits for BMW, document the changes in the computing environment and tells about future plans.

Figure 5: Automobile crash simulation at BMW using GRD

2.1 Starting Point and Initial Resource Management Requirements

The BMW Crash department EK20 performs compute-intensive crash simulations with PAM-CRASH. The results are heavily used in the design process of the vehicles. Back in 1994 the environment was very simple - computational work was done on 10 CPUs by 3 engineers. Users resorted to “phone call scheduling”, which was not a problem with just a few users.

The department was growing rapidly, and “phone call scheduling” was no longer possible. A second compute server was installed; thus it was necessary to provide transparent access for users. BMW identified this as being a crucial problem with multiple servers. The crash

department wanted the CPU load to be equally distributed without any servers being overloaded. Also monitoring capabilities of resource management systems were considered as important so as to being able to see at a glance if something is wrong with a machine.

2.2 Current Resource Management Status and Achievements

CODINE was installed at a very early stage of automated job distribution. Therefore, the BMW crash department never experienced problems in managing the enormous growth of the cluster. The department grew by a factor of 10. The number of CPUs used now is about 370 on 11 compute servers and more than 100 workstation - still increasing. The applications have become more complex, the number of test cases and the variations per model have grown exponentially. BMW feels that they would not have been able to manage this growth without the help of CODINE/GRD.

Today, about 45 users submit jobs to the cluster, utilization is far more than 80% of the total CPU time, downtimes included. CODINE/GRD has optimized the utilization of the machines and thus reduced new purchases to a minimum. So there is a direct connection to savings.

Counting only the reduction of purchases, BMW estimated savings of more than 1 Million US$. As an indirect consequence, there was a speedup in simulations, and the crash department engineers were able to focus on their scientific business instead of IT processes. In summary, development cycles for BMW products were shortened.

Current activities focus at making use of the dynamic resource utilization management facilities of GRD. It allows a flexible priority enforcement according to a project’s budget. Furthermore, an additional increase in utilization to an overall 90% is expected. This is being approached by starting low-priority jobs during the day. During the night, when the urgent tasks have finished, these jobs will gain more priority.

2.3 Future Plans

The BMW crash department plans to use hundreds of workstations at night and on weekends in addition to the compute servers. GRD has the potential to keep the necessary administrative efforts low since the simulation jobs must consider the individual workstation hardware and software (memory, CPU, operating system) and GRD accommodates these requirements.

3 GRD at ARL

The Army Research Lab (ARL) has developed a state-of-the-art Major Shared Resource Center (MSRC). It is now in full production, providing leading HPC technologies and tools to aid DoD scientists and engineers, according to Charles J. Nietubicz, Director of the ARL MSRC and Chief of ARL's High Performance Computing Division.

Of the four major centers in the DoD High Performance Computing DoD modernization program, ARL MSRC is the primary provider of multi-architecture classified systems. A full complement of unclassified HPC systems, as well as state-of-the-art near-line mass storage, and leading edge scientific visualization.

Recently ARL MSRC took delivery of two Cray T916, each configured with 8 processors and 512 Megawords (4Gb.) of memory. The addition of the most powerful parallel vector processing available to DoD augments the SGI distributed shared memory Origin 2000s,

SGI/Cray J932s, and SGI Power Challenge Array serving defense scientists and engineers using the center.

3.1 Key enabling technology: Queuing Up for GRD

GRD, as a new queuing system that dynamically allocates resources to the users according to preset 'sharing' guidelines and allows control of resources from a single point, is in use at the ARL MSRC since 1997. It dynamically adjusts available resources based on a share tree structure and the jobs currently running.

The system is a first for MSRC in that it controls all machines from a single queuing environment. ARL considers this as being “[...] a very good first step towards area-wide queuing for the MSRCs” according to Denice Brown, Manager of Operations and Customer Services at the Center.

3.2 How GRD is Configured at ARL

At ARL, all users submit their jobs from a single interface no matter what machine the job is going to. In the present share tree structure, 20% of system resources are allocated to the DoD Challenge projects. The remaining 80% is allocated as follows: Army (30%), Navy (30%), Air Force (30%) and other DoD agencies (10%). This is in contrast to most traditional systems, which run jobs according to an assigned priority and size, and which cannot be adjusted once they begin execution.

Approximately 500 engineers, researchers and developers directly from ARL and other locations use the GRD system. Typically, user submit sequential as well as parallel jobs, the latter mainly using the Message Passing Interface (MPI) standard.

Under the GRD system, users specify information about the type of job they want to run. The system will automatically assign the job to the appropriate queue based on available resources and the job originator's position in the tree. Other parameters such as other jobs currently running for the same user are also factoring in, so that no one user can monopolize system resources. However, if necessary, the system administrator can override the automatic structure.

In addition, some applications such as certain periodic weather forecasts have increased priority as compared to other work, even though the departments or projects to which they belong may have little actual resource share.

Therefore, ARL´s resource utilization policy requirements can be summarized as:

- Fair distribution of resources over a sliding time window
- Short term over-commitment of resources for the price of later compensation
- Express & dead-line jobs
- Automatic enforcement of policies
- Manual override capabilities

3.3 Major Achievements and Future Directions

In more than a year of operation GRD has successfully implemented ARL´s aforementioned operational goals as can be seen from the quote contained in figure 6.

In the future, ARL is looking into the direction of even more sophisticated share distribution schemes, wider usage of distributed memory applications in the GRD context and area-wide queuing activities.

Denice Brown - Mgr. of Operations & Customer Services

The user can be assured of getting his or her job run in the most fair and efficient manner according to set sharing policies;

... the system administrator has the benefit of having the resource allocation done automatically as well as being able to monitor resources at both the site and individual job level."

Previously, resources were **degraded** for everyone when a few users **"overused"** the system.

Figure 6: ARL quote

4 Conclusions

The two sample customers scenarios of BMW and ARL as well as the technical description in the first section of this paper documented that GRD is a novel resource management system product with outstanding features targeted for HPC centers, enterprise or large department computing facilities as well as application or computing service providers.

As opposed to traditional queuing systems it has several unique capabilities, such as automated implementation of resource utilization policies across an enterprise computing environment combined with support for heterogeneous environments and flexible administration.

GRD helps site managers to gain overview on the resource utilization profile of an enterprise, to distribute resources fairly and to implement level-of-service agreements.

5 Bibliography

[ARL97] *link*, The ARL MSRC Newsletter, Volume 1, Issue 1, Fall 1997.

[BAKE95] Mark Baker, Geoffrey Fox, and Hon Yau, *Cluster Computing Review*, Technical report CRPC-TR95623 of the Center for Research on Parallel Computation, Rice University, November, 1995, 70 pages.

[CASA96] Christine Casatelli, *Managing in the Distributed World*, RS/Magazine, April 1996, Vol. 5, No. 4, pp 30-38.

[COD197] *CODINE Installation and Administration Guide*, Version 4.1, 143 pages., 1997.

[COD297] *CODINE User's Guide*, Version 4.1, 81 pages., 1997.

[FISC95] Craig Fischberg et. al., *Using Hundreds of Workstations for Production Running of Parallel CFD Applications*, Proceedings of Parallel CFD '95, Elsevier Science Publishers B.V., The Netherlands, 1995, 14 pages.

[GEN98] *README!*, GENIAS bi-monthly newsletter, June 1998.

[GILL96] Philip J. Gill, *Integration Alters Systems Management Mix*, Software Magazine, Jan 1996, pp. 65-71.

[GRD197] *GRD Release Notes & Installation Guide*, Version 1.1, 103 pages, May 1997.

[GRD297] *GRD User´s Guide*, Version 1.1, 111 pages, May 1997.

[IEEE94] IEEE Standard 1003.2d-1994, Portable Operating Systems Interface (POSIX®) Part 2: Shell and Utilities, Amendment 1: Batch Environment.

[JON97] Jones, J. P., *Brickell, C., Second Evaluation of Job Queuing/Scheduling Software*, NASA Ames Research Center, U.S.A., NAS-97-013, June 1997.

[KAPL94] Joseph A. Kaplan, Michael L. Nelson, NASA Langley Research Center Report #109025, June, 1994, 48 pages.

[KAY88] J. Kay and P. Lauder, *A Fair Share Scheduler*, Communications of the ACM, January 1988, Vol 31, No 1, pp 44-55.

[ROCH96] Larry Roche, Henry Newman, Johannes Grawe, *Operational Needs for Software Products That Manage Batch Workloads and Distributed Computers*, Proceedings of Computer Measurement Group, CMG 96, San Diego, CA.

[WALD94] Carl A. Waldspurger and William, E. Weihl, *Lottery Scheduling: Flexible Proportional-Share Resource Management*, Proceedings of the First Symposium on Operating System Design and Implementation, November 1994, 11 pages.