Scalable Computing: Practice and Experience
Volume 10, Number 2, pp. 189–199. http://www.scpe.org
ISSN 1895-1767

THE INFLUENCE OF THE IBM PSERIES SERVERS VIRTUALIZATION MECHANISM ON DYNAMIC RESOURCE ALLOCATION IN AIX 5L

MACIEJ MLYNSKI*

Abstract. This paper presents analysis of the influence of the virtualization mechanism of IBM pSeries servers on dynamic resource allocation in AIX 5L operating system. It raises issues of economic use of resources and distribution of processing power. Some innovative solutions are proposed as methods for dynamic resource allocation. These are: micro-partitioning and partition load manager utility. Some results of experiments are presented. The goal of these experiments was to estimate the influence of selected AIX 5L operating system adjustments on computation efficiency.

Key words: virtualization, AIX, DLPAR, dynamic resource allocation, PLM, VIO, hypervisor, micro-partitioning

1. Introduction. At present, the demand for server processing power begins to go beyond its processor capabilities. However, installing more processors and memory causes the resources to become more expensive [3, 14, 15]. Therefore research is being undertaken to investigate how to utilize the resources more efficiently. In this direction important research are grid and autonomic computing [3, 12, 17]. During the computer systems design phase we need to consider installing more than one application and database onto one individual server [2].

Many factors can influence multiple applications, such as splitting of mission critical applications and development, creating an environment for patch installation as well as computer systems for many different customers. A multi-instance environment could also be motivated by efficiency reasons [4, 5, 6]. Efficiency would also influence a *multi-instance* environment. Unfortunately, several conflicting instances of databases or applications on one server may create a problem with conflicting processing power resources. The problems can be resolved by adequate parameters modifications inside the operating system or server microcode.

The dynamic resource allocation mechanisms are based on well documented theory concerning mainly elements of the operational research such as queuing and resource management theory, Markov chains, analysis of time series and gradient methods [9, 16].

The paper discusses efficient use of resources and distribution of processing power in IBM pSeries servers. Three solutions for the improved dynamic resource allocation are being discussed: micro-partitioning, capped and uncapped processing power as well as the partition load manager utility. Results of experiments are presented, and the influence of virtualization technology on IBM pSeries server efficiency are analysed. The following are the main contributions of the paper:

- Efficiency problems in IBM pSeries servers have been experimentally analysed. The experiments have been done at customer sites in banks and others complex installations.
- Implementation of an efficient policy for partition load manager utility has been done.
- The set of virtual machine parameters has been analysed in concern with dynamic modification ability.
- It has been proved that to improve entire hardware efficiency by dynamic changes of the amount of assigned memory and the number of virtual processors, selected parameters in the operating system should be changed as well.

Dynamic resource allocation in operating systems involves a huge amount of parameters, but only their correct adjustment can give the expected results. The set of experiments presented in section four shows that increasing of processing resource in logical partition not always improve the overall system efficiency.

The goal of this paper is to indicate possibilities of improvement of dynamic resource allocation characteristics. In particular the attention was turned onto mechanisms of tuning operating systems and tuning the I/O subsystems.

2. The virtualization and workload management on AIX. A progress in methods for data transmission and information processing enforces changes in computer systems architecture. In the AIX 6.1 operating system modern virtualization mechanisms have been introduced. The mechanisms enable creating flexible configurations by administrators and systems architects. In the IBM pSeries servers environment *micro-partitioning* (dividing resources into small pieces), *IVE (Integrated Virtual Ethernet)*, *Shared Ethernet*, and *virtualization* have just been introduced.

*ASpartner Sp. z o.o., ul. Jaworowa 5, 05-816 Michalowice, Poland, (m.mlynski@aspartner.com.pl).

Conception of virtualization mechanism in IBM pSeries servers is similar to the concepts implemented in other virtual environments such as Xen and VMWare [19]. The conception assumes that physical resources like processors units, memory, I/O drawers and adapters can be shared. Fig. 2.1 shows the scheme of logical partitions on IBM pSeries architecture. [1] The new computer system architecture has introduced a number of new elements, which system architects should consider when designing information systems.

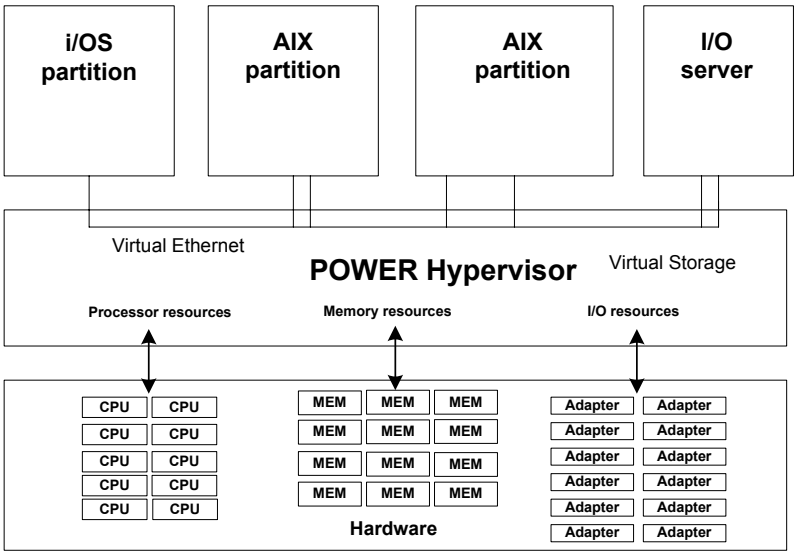

FIG. 2.1. *Scheme of logical partitions on IBM pSeries Systems server architecture. [1, 15]*

Another subject on which emphasis is placed today is dynamic resource allocation issue. Since not long time ago we know the *Work Load Manager* functionality on AIX operating system [8, 13, 14]. Now we can also use a new functionality called DLPAR (dynamic logical partitions). The functionality allows for on-line resources allocation between logical partitions e.g. processors, memory and I/O. Introduction of the DLPAR is possible because new servers have an intermediate layer called a *hypervisor* [7]. In this solution, memory is not addressed directly by the operating system. The hypervisor is serving a translation buffer role. Therefore these solutions allow for the use of a tool called *Partition Load Manager*. They also allow dynamic hardware manipulation. As a result, it is possible to configure a computer system so, that if required, we can dynamically add processors and memory to a particular partition. Virtualization technologies provide also abilities to switch resources between servers like WPARs (Workload partitions) which provides a single focal point for management of AIX 6.1 across an enterprise and *Live Application Mobility* which allows on-line realocations between servers. Hardware *hypervisor* is supporting a few kinds of operating systems installed together on one physical machine. On the *IBM pSeries System* hardware platform these operating systems, which can be installed in that way are *AIX*, *i/OS*, and *Linux* (*SLES*, *RHEL*). These operating systems can use shared and virtual resources.

In the partitioned environment, memory utilization is described by two parameters: the amount of allocated memory and the number of scans per second. Therefore, a partition load manager has its own knowledge database. In the database, the partition load manager stores information about system utilization. Resource allocation is realized on the basis of stored data and defined policy rules.

The process of making policy rules must be consistent with AIX specification. Memory space in AIX is divided into many sub spaces e.g. *working*, *client* and *persistent*. The working memory is a memory, which is used by program codes and libraries. Therefore, the client and persistent spaces are dedicated to storing data. How much memory to be assigned for a file cache and how much for a program code is adjusted by the *minperm* and *maxperm* parameters. On the basis of these parameters, the *page stealer* will decide which memory pages should be released.

Resource allocation between logical partitions is governed by the following rules: the hardware management console is sending a request to AIX for their release. After that, the operating system releases the resources

and sends information to the hardware management console that the resources were released. Fig. 2.2 shows a schematic of connections of hardware management console with particular logical partitions, partition load manager and resource manager. [1]

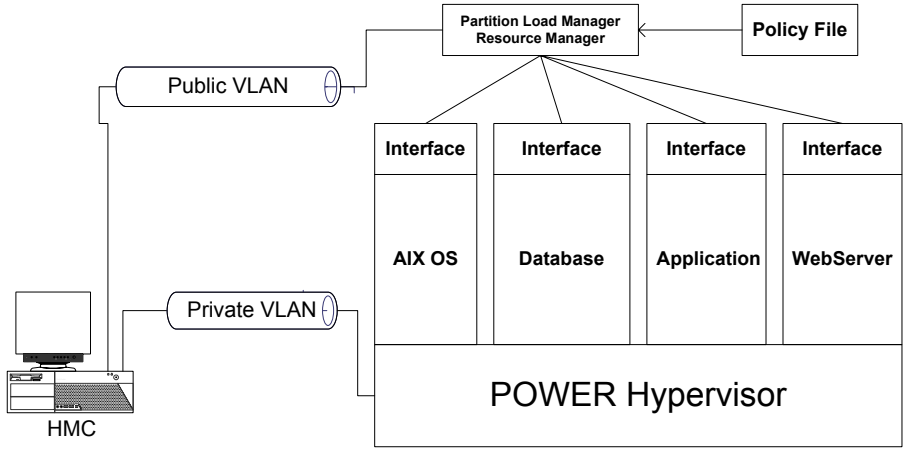

FIG. 2.2. *The scheme of physical connections to partition load manager. [1, 15]*

The policy rules of partition load manager must estimate and define minimum, maximum and guaranteed amount of resources, priority, weight and thresholds for every partition. Administrator should also define minimum and maximum thresholds when events will be generated for every resource. If the same partition sends requests for new resources, the system will release them according to the following rules:

- Resources available and un-assigned,
- Resources-allocated but not used,
- Resources from partitions with lowest priority.

From the above we can draw the conclusion, that the architects must plan the resources allocation before or during the design of a system, to ensure efficient resource allocation.

The design of policy rules has to respect both the architecture of a computer system and business application features. We have several system perspectives which should be respected on the basis of the P6 architecture, e.g.:

- Security, *CAPP/EAL4+* (controlled access protection profile and evaluation assurance level 4+)
- Architecture of central processing units e.g. *simultaneous multi threading*, *memory affinity*, *L3* sharing between processors,
- *RAS* architecture (reliability, availability and scalability),
- *Virtual* and *shared I/O*,
- *Virtual* and *shared Ethernet*,
- Capacity on Demand,
- Multiple operating systems.

Sometimes business applications and databases do not allow dynamic changes. Therefore, we must remember that applications and databases should allow for dynamic resource allocation changes and applications should be able to send requests to the hardware management console with demand for new resources. Applications and databases should also be able to reduce their own resources if the hardware management console needs it.

3. Policy rules for Partition Load Manager. If we want to create policy rules for a partition load manager we must first create new resources. In our environment, we have created two resource groups. For each of these resources groups we have defined the maximum and minimum amount of resources to be need. During this process we can also indicate if the processors are to be dedicated or shared. Fig. 3.1 shows the defined resources.

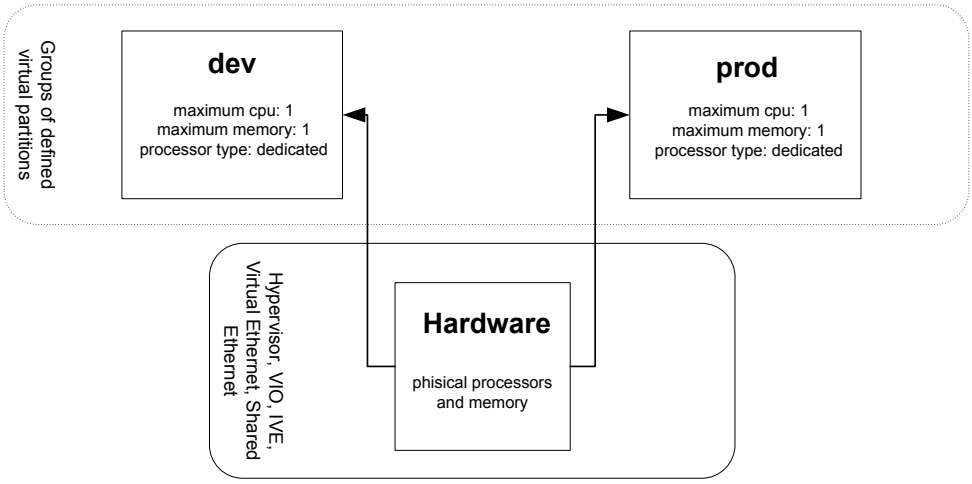

FIG. 3.1. *Resource group in Partition Load Manager.*

The *Partition Load Manager* has some parameters to define thresholds indicating when resources will be allocated between logical partitions. Frequent physical changes can cause low system stability. However, setting the thresholds to high values can cause that resources will never swap.

Next, we have assigned logical partitions for particular physical group. During that, we had to determine the *maximum*, *minimum* and *guaranteed resources* for all logical partitions. This operation concerns processors and memory.

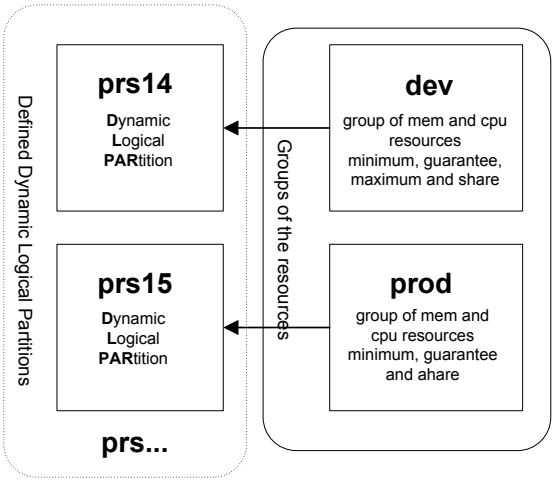

FIG. 3.2. *The definition of logical partitions in Partition Load Manager.*

Two logical partitions for this experiment were created. The first was called prs1 and had 1 CPU and 4 GB RAM. The second logical partition was called prs14 and had 4 CPU and 6 GB RAM. Fig. 3.2 shows the scheme in which one group is assigned to one logical partition. The logical partition called prs15 is assigned to a production group. Therefore the logical partition called prs14 is assigned to the development group.

For policy rules as well as a description of logical partitions, parameters that determine the behavior after crossing of thresholds should be specified. The essential parameter is a parameter which determines whether a partition load manager should release free resources automatically. In the case of a request for more resources to be sent by logical partition, the partition load manager will firstly give resources not yet allocated.

TABLE 3.1
Parameters of the logical partitions.

	Instance	Group	Defaults
Entitled capacity delta	10	-	10
Memory delta	1	1	1
CPU notisy intervals	5	6	6
Memory notisy intervals	5	6	6
CPU load average high threshold	0.9	1.00	1.00
CPU load average low threshold	0.1	0.40	0.40
Minimum entitlement per VP	0.4	-	0.50
Maximum entitlement per VP	0.4	-	0.50
Memory utilization high threshold	50	50	90
Memory utilization low threshold	90	90	50
Memory page steal high threshold	0	0	0
Immediate release of free CPU	yes	no	yes
Immediate release of free memory	yes	no	no

In cases when there are fewer resources, some should be released from the running partition. Then, they are released on condition that the logical partition is using more resources than was guaranteed. If all logical partitions need their own resources, then the partition load manager decides which resources should be released on the basis of partition weight. The weight of a logical partition is specified during the server partitioning.

Table 3.1 shows the parameters of a logical partition. All logical partitions can have different parameters. However, the group restrictions should be valid and proper i. e. precise match with expected requirements. Also tunable resources entitlements should be defined. Entitlements are a kind of resources limits.

Resources are re-allocated mainly based on these parameters. Resources between logical partitions are distributed in harmony with defined maximum, minimum and guaranteed values for memory and processors. In P5 and P6 environment it is possible to change the CPU allocation by 0.01 of their processing power. More precise explanation of the tuning parameters can be found in Table 3.2.

After the policy rules definition, we can run partition load manager. In our test environments, the system parameters were changed many times. After running the partition manager we were able to show statistics. Fig. 3.3 shows memory statistics for a logical partition prs15. Fig. 3.4 shows the processor utilization. Processing statistics show average processor utilization.

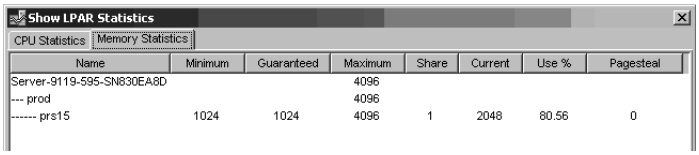

FIG. 3.3. *Memory statistics for logical partitions.*

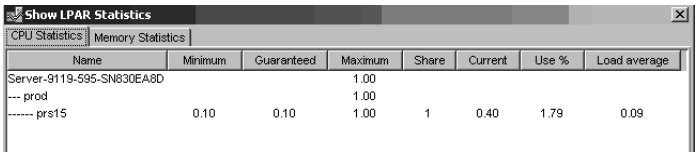

FIG. 3.4. *The CPU statistics for logical partition.*

The partition load manager can be run in passive or active mode. In passive mode only the analysis of policy rules is possible. The next stage, after analyzing the partition load manager, is to be run in active mode. In this case, the dynamic resource allocation is possible.

Not all applications support dynamic resource allocation. Applications which do not support dynamic resource allocation often use bind processors and use *pinned memory*. *Pinned memory* is a memory whose segments are placed always at the same address and can not be swapped into the paging space. *Pinned memory* is used by subsystems of databases like *aio* (asynchronous I/O). In such applications, dynamic resource allocation

TABLE 3.2
CPU-related parameters [2].

Tunable	Description
Entitled capacity delta	The amount of CPU entitled capacity to add or remove from a shared processor partition. The value specifies the percent of the partition's current entitled capacity to add or remove.
CPU notify intervals	The number of 10 second sample periods that a CPU-related sample must cross a threshold before the reallocation will take effect.
CPU load average high threshold	The processor load average high threshold value. A partition with an average load above this value is considered to need more processor capacity.
CPU average load low threshold	The CPU average load low threshold value. A partition with a average load below this value is considered to have unneeded CPU capacity.
Minimum entitlement per virtual processor	The minimum amount of entitled capacity per virtual processor. This attribute prevents a partition from having a degraded performance due to too many virtual processors relative to its entitled capacity. When entitled capacity is removed from a partition, virtual processors will also be removed if the amount of entitled capacity for each virtual processor falls below this number. Default value is 0.5. Minimum value is 0.1. Maximum value is 1.0.
Maximum entitlement per virtual processor	The maximum amount of entitled capacity per virtual processor. This attribute controls the amount of available capacity that may be used by an uncapped (uncapped means that amount of resources is not strictly limited) shared processor partition. When entitled capacity is added to a partition, virtual processors will be added if the amount of the entitled capacity for each virtual processor goes above this number. Increasing the number of virtual processors in an uncapped partition allows the partition to use more of the available processor capacity.
Immediate release of free CPU	Indicates when processor capacity not needed by a partition is removed from the partition. The value of no indicates unneeded CPU capacity remains in the partition until another partition has a need for it. The value of yes indicates unneeded CPU capacity is removed from the partition when the partition does not need it any longer.

can cause a crash. The physical resource allocation have also a big impact on system efficiency. The strength of the influence it has on I/O subsystems is discussed in the following paragraph.

4. The analysis of the influence of disk subsystem parameters on operating system efficiency. We distinguish sequential and random disk transfers. In random disk transfers, administrators do not have much freedom to tune it, but in sequential transfers there is a bigger possibility [10, 11]. The most important parameters are *minpagereadahead* and *maxpagereadahead*. How the mechanism works is shown in the following examples, assuming that *minpagereadahed=2* and *maxpagereadahead=16*.

For this setup, in the case of sequential reading, first transfer from disk will read two pages. If system recognizes that the pages, which were read are sequential, the next transfers will read 4 pages, next one 8, 16 etc. All other transfers will read 16 blocks in one reading cycle. A maximum size of transfer is always equal to *maxpagereadahead* value. For the system to be efficient, it should have enough memory for every read. The system should have *maxpagereadahead* free pages. How many free pages will be adjusted by *minfree* and *maxfree* tuning parameters? The number of free pages should never fall below *minfree*. If the numbers of free pages reach *minfree*, then the *page stealer* process will release pages until the amount equals *maxfree*. *Maxfree* value is defined by the following formula:

$$maxfree = minfree + maxpgahead \tag{4.1}$$

TABLE 4.1
Influence comparison of system values on their efficiency.

	Min free	Max free	Max page read ahead	Mem pools	File size	Trans. type	Time
1	960	1088	8	1	1GB	read	1.14s
2	960	1088	16	1	1GB	read	1.07s
3	960	1088	32	1	1GB	read	1.01s
4	960	968	8	1	1GB	read	1.10s
5	960	968	16	1	1GB	read	1.15s
6	960	968	32	1	1GB	read	1.18s
7	960	1088	8	1	1GB	r/w	20.49s
8	960	1088	16	1	1GB	r/w	16.44s
9	960	1088	32	1	1GB	r/w	17.60s
10	960	968	8	1	1GB	r/w	19.16s
11	960	968	16	1	1GB	r/w	19.45s
12	960	968	32	1	1GB	r/w	20.02s

The minfree value is determined by experiments. It is usually equal to the size of one of system processes, which should be run fast. We must remember that the memory below *minfree* value is wasted. The experiment was carried on in highly loaded computer systems. To carry out the experiment, one should also have correction on the number of processors and memory pools. After the corrections, the *maxfree* is defined by the following formula:

$$maxfree = minfree + (maxpgahead * nCPU) \tag{4.2}$$

For small computer systems, the *mempool* value is equal one. However, in quite big computers, the value is adjusted during an installation process. For example, in one of tested computers that has 40 GB RAM, the number of *memorypools* was equal 5. The value can be changed. After a correction for *mampool*, the *maxfree* will be defined by the following formula:

$$maxfree = minfree + (maxpgahead * nCPU/mempools) \tag{4.3}$$

From this formula we see that during the dynamic memory allocations we should also modify the *Virtual Memory* values. The latest implementations of AIX require to restart the operating system after changes of the *memools* value. In the future, when we change the *mempools* value without rebooting, the *mempools* value should be changed automatically during dynamic resource allocations. To check how big influence on system efficiency the parameters of *maxfree*, *minfree* and *maxpagereadahead* have, experiments were carried out. During the experiments some files were copied. The experiment was carried out on one of logical partition on p595 server. The logical partition has 0.4 CPU (POWER5 1.9 GHz) and 2047 MB RAM. In Table 4.1 the results have been presented.

The experiments (1-6) confirm that an increase of *maxpagereadahed* increases system efficiency when we have enough free memory for pages to be allocated. The second part of this table shows the results of parallel read and write. The experiments 7-12, show that when we have small number of pages, the increase of *maxpagereadahead* value results in longer files copying time. It means that we should be careful about memory free pages. System will start running the *memory stealer* process when we do not have enough free pages, and so, it will result in weak system efficiency.

To better understand the dynamic resource partitioning another test was executed. The experiment was done on the p5 server with 11 virtual partitions and virtual VIO server. All eleven virtual partitions had no physical I/O adapters. All physical I/O were connected to the VIO server. The physical resources were divided and exported to particular partitions. Processing power was divided in *shared* and *uncapped* mode i. e. all partitions could use all CPU resources. The amount of utilized memory for each partition could be settled dynamically between 1 and 16 GB.

To measure the I/O efficiency the following tests have been executed: a) write the data to disk; b) read from disks; c) dynamic reallocating memory to particular partition during I/O operation.

The disk efficiency experiment showed that maximum transfer rate for the I/O for one partition was 320 MB/sec. During the experiment the 1MB block size was used. In Fig. 4.1 we can see partition utilization as a

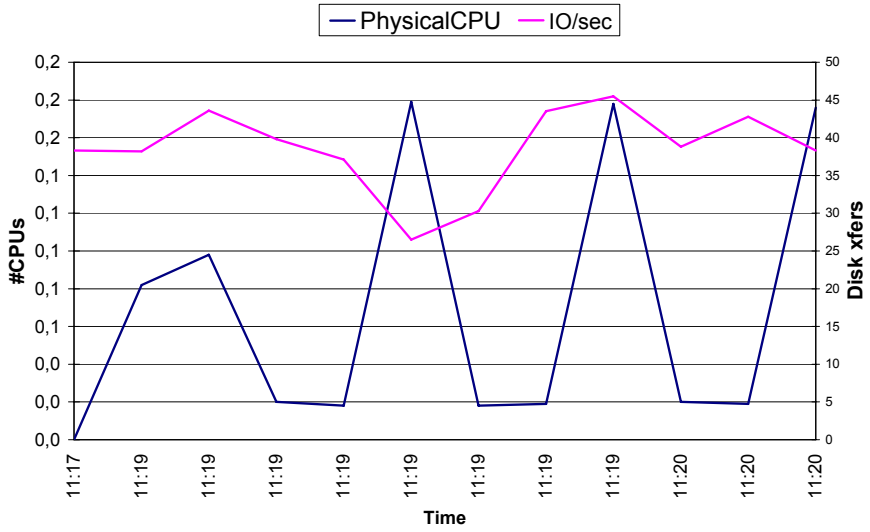

FIG. 4.1. *Lpar utilization during disk writing.*

function of time. Most of processing power spent on communication between external storage and the server was used by Virtual I/O server. Therefore on the LPAR only 0.2 CPU was used.

The second experiment showed the system behavior during dynamic memory reallocating. For one of partitions, the amount of memory was increased from 1 GB to 5 GB. As we can see in Fig. 4.2, the percentage of memory usage was decreased slightly. The dynamic memory reallocation works in the following way: the hardware management console is communicating with the hypervisor, if there is free memory than allocate it, then HMC is sending the message to operating system which information containing the memory address that might be used. Fig. 4.3 shows the same experiments which are shown in Fig. 4.2. The The numbers of *forks* and *execs* decreased in Fig. 4.2 at the same time when the amount of used memory changed in Fig. 4.3. For current version it is possible to decrease, increase and relocate memory simultaneously between two partitions. When decreasing an amount of memory from a partition all applications must release particular memory segments.

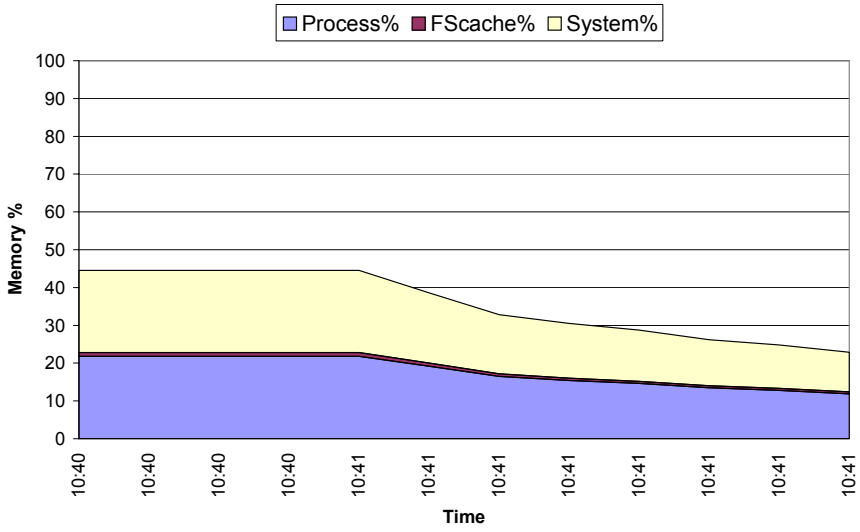

FIG. 4.2. *Memory usage.*

More memory caused that the amount of forks and exec slightly decreased, also the numbers *pswitch* and *syscalls* were decreased automatically during the memory allocation.

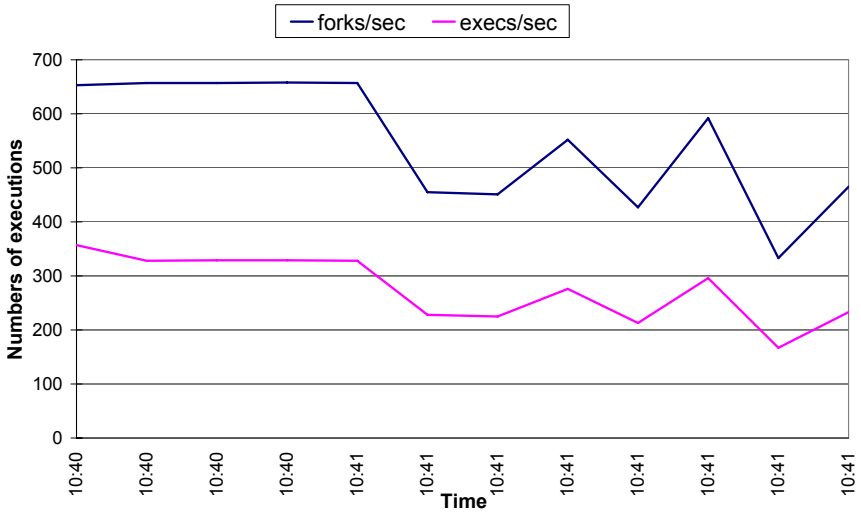

FIG. 4.3. *Processes.*

The system statistics show that when the amount of memory was increased the utilization of kernel read/write system calls, *hypervisor poolidle* has decreased. The parameters show that the capacity of the partition should be better and disk transfers should increase. Then it shows that the disk transfers slow down after the dynamic changes.

Next experiments show the system behavior when the numbers of processors and amount of memory has been changed. These experiments were taken in P5 server with 1.6MHZ CPUs with *uncapped* mode. The external devices were the following: (1) external storage DS4700 (2) logical volume created on RAID5 (3) fiber channel switches with 4Gbps adapters. Tests were generated by using the tools from *iozone* [18]. The first experiment has been done by using 0.1 CPU and 1GB RAM. The results are in Fig. 4.4.

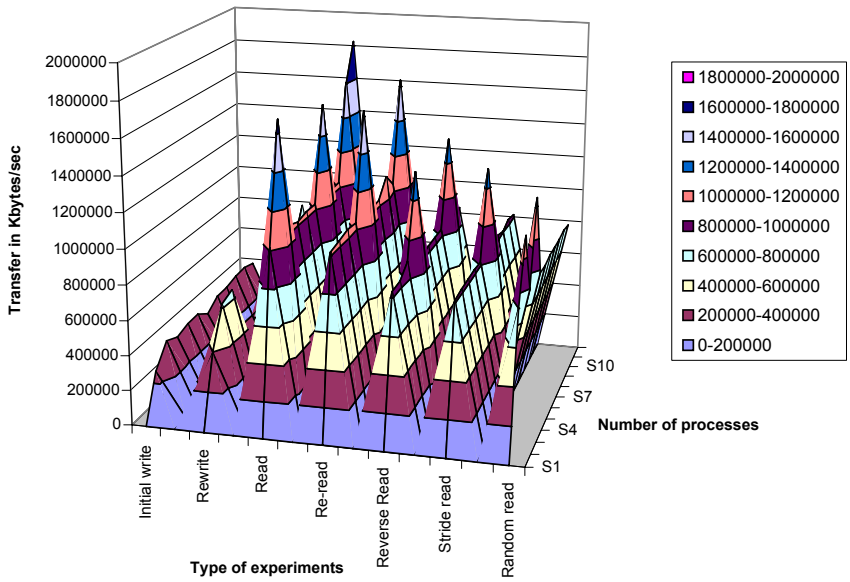

FIG. 4.4. *Transfers with 1GB 0.1 CPU.*

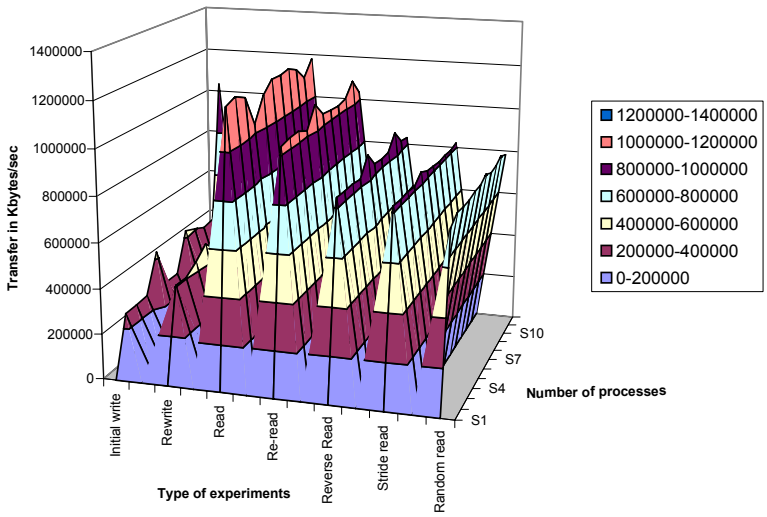

FIG. 4.5. *Transfers with 6GB and 3CPU.*

Because the intention of the experiments was to show an environment similar to a real system, the figures show series of experiments such as read, write, re-read, reverse read, stride read and random read. All of the tests have been done for various ranges of methods i. e. write, read, re-read, reverse read. The initial setup of this program was the following: minimum of processes equal 10, the maximum of processes equal 20. The generated plot (Fig. 4.4) shows S1 as the minimum numbers of processes, S10 as the maximum. Second experiment (Fig. 4.5) in the series has been done with 3 CPUs and 6 GB RAM.

The above experiments have shown that a logical partition with more memory and processing power (Fig. 4.5) has the ability to proceed with the I/O requests in a more stable way, because the transfer rate is the same for 10 and 20 processes. The Fig. 4.4 shows that average transfers are worst and also some undeterminism has been found. Because the transfer rate for read reached 1800000 kBytes/sec this is more than for a read with 6 GB RAM i. e. 1200000 kBytes/sec.

The experiments have shown that the influence of processor frequency on system performance is less important now than it was many years ago. For high frequency processors system parameters describing process scheduling have less influence on performance, because the parameters describing access to memory and storage are more important for this purpose these days.

5. Conclusions. Considering the fact that computer science and computer parallel systems architecture are still evolving, the problem of tuning dynamic resource allocation is a novelty. Continuously changing environment enforces the dynamic resource allocation mechanisms to improve.

This article concerns innovative techniques like *virtualization* and *partition load management*. Thanks to them, we can dynamically manage processing efficiency. The mechanisms make it possible to dynamically move resources between logical partitions on a server. Therefore the computer systems design can be more adequate to demands of new applications and databases. Consequently, we can say that it is possible to prepare methods for tuning operating system properties in order to improve suitability for dynamic resource allocation.

A reduction of the resource pool during normal system operation requires important changes in the operating system tuning values. The conclusion of the analysis and experiments is that in this respect even small changes in hardware can have big influence on system efficiency.

REFERENCES

[1] M. MLYNSKI, *The analysis of influence of IBM pSeries servers' virtualization mechanism on dynamic resources allocation in AIX 5L*, Proceedings for IEEE CS International Conference ISPDC, 2008.

[2] A. B. BLANK, M. GIEPARDA, J. HAUST, O. STADLE, D. SZERSI, *Advanced POWER Virtualization on IBM @server p5 Servers: Introduction and Basic Configuration*, IBM ITSO, Austin, 2005.

[3] *Autonomic Computing*, `http://www.research.ibm.com/autonomic`
[4] P. Bari, P. Cassier, A. Defendi, J. Hutchinson,A. Maneville, G. Membrini, C. Ong, *System Programmer's Guide to Workload Manager.*, IBM ITSO, Austin, 2005.
[5] P. Bari, D. Dillenberger, H. Morrill, *System Programmer's Guide to Workload Manager.*, IBM ITSO, Austin, 2005.
[6] T. Cederlof, A. Klark, T. Herlin, T. Ostaszewski, *IBM Certification Study Guide AIX Performance and System Tuning.*, IBM ITSO, Austin, 2000.
[7] A. Dembeter, S. Dutta, A. Roll, S. Son, *AIX 5L Differences Guide Version 5.3 Edition.*, IBM ITSO, Austin, 2004.
[8] D. Clitherow, S. Herzog, A. Salla, V. Sokal, Trethewey J., *OS/390 Workload Manager Implementation and Exploitation.* , IBM ITSO, Austin, 2004.
[9] T. Czachorski, *Analitical Queuing Model for Performance Evaluation of Computer Systems and Computer Networks.*, ProDialog, no. 16, Poznan 2003.
[10] *IBM AIX Technical Documentation: Performance Management Guide.*, *VMM page replacement tuning.*, 2008.
[11] *IBM AIX Technical Documentation: Performance Management Guide.*, *Tuning Logical Volume Striping.*, 2008.
[12] J. Jann, L. M. Browning, R. S. Burugula, *Dynamic reconfiguration: Basic building blocks for autonomic computing on IBM pSeries server.*, IBM Systems Journal, vol. 42, no. 1, 2003 Autonomic Computing.
[13] M. Mlynski, *Dynamic Resources Allocation in AIX 5L, Real Time Systems - Design and Applications.*, XII Conference of Real-Time Systems , Ustron 2005.
[14] M. Mlynski, *Analysis of Using An AIX Dynamic Resource Allocation Mechanism to Describe a Utility Level of Server in Oracle Data Bases Environment*, Studia Informatica, vol. 26 no. 3(64), Gliwice 2005.
[15] C. Matthys, M. Mlynski, N. Tollet, G. Barbati, H. Chauhan, B. Dierberger, R. Marchini, H. Wittmann, *Planning, Installing and Using the IBM Virtualization Version 2.1*, IBM ITSO Poughkeepsie (USA)2006.
[16] S. Wegrzyn, *Informatics as Specific Domain on Motion and Processing of Information.*, ProDialog, no. 16, Poznan 2003.
[17] S. Wegrzyn, *Resource Management in Grids.*, ProDialog, no. 16, Poznan 2003.
[18] *IOzone*, `http://www.iozone.org`
[19] G. Vallee, T. Naughton, L. S. Scott, *System Management Software for Virtual Environment.*, Proceedings of the 4th international conference on Computing frontiers, 2007.

Edited by: Marek Tudruj, Dana Petcu
Received: February 7th, 2009
Accepted: June 28th, 2009