EnergyScale for IBM POWER6 microprocessor-based systems

H.-Y. McCreary
M. A. Broyles
M. S. Floyd
A. J. Geissler
S. P. Hartman
F. L. Rawson
T. J. Rosedahl
J. C. Rubio
M. S. Ware

With increasing processor speed and density, denser system packaging, and other technology advances, system power and heat have become important design considerations. The introduction of new technology including denser circuits, improved lithography, and higher clock speeds means that power consumption and heat generation, which are already significant problems with older systems, are significantly greater with IBM POWER6™ processor-based designs, including both standalone servers and those implemented as blades for the IBM BladeCenter® product line. In response, IBM has developed the EnergyScale™ architecture, a system-level power management implementation for POWER6 processor-based machines. The EnergyScale architecture uses the basic power control facilities of the POWER6 chip, together with additional board-level hardware, firmware, and systems software, to provide a complete power and thermal management solution. The EnergyScale architecture is performance aware, taking into account the characteristics of the executing workload to ensure that it meets the goals specified by the user while reducing power consumption. This paper introduces the EnergyScale architecture and describes its implementation in two representative platform designs: an eight-way, rack-mounted machine and a server blade. The primary focus of this paper is on the algorithms and the firmware structure used in the EnergyScale architecture, although it also provides the system design considerations needed to support performance-aware power management. In addition, it describes the extensions and modifications to power management that are necessary to span the range of POWER6 processor-based system designs.

Introduction

The next-generation server-class processor in the IBM Power Architecture* technology-based family, the IBM POWER6* processor, offers significantly higher clock frequencies and uses improved lithography with smaller device sizes than its predecessors. As processor power and heat increase and components are packed ever more tightly, they increase not only in performance but also in their power consumption and heat generation, compounding what was already a thermal challenge. As a result, traditional high-margin designs can no longer take full advantage of the technology advances provided by POWER6 processor-based systems. Active measurement and management of power and thermal system attributes is required to provide power efficiency and maximum performance for critical workloads. The implementation of performance-aware power and thermal management for POWER6 processor-based systems is called the *EnergyScale* architecture*.

The EnergyScale architecture meets a number of basic customer requirements for system-level power management, as described below.

Power and temperature data collection and reporting

Users and data-center operators need to know how much power a system draws and how much heat it generates.

0018-8646/07/$5.00 © 2007 IBM

Traditionally, such data is collected only during crises by using external measurement devices. The EnergyScale architecture meets this requirement by providing continuous data collection using built-in sensors and firmware.

Power capping and power limitations

System designers deal with power limitations imposed by constraints on packaging and power supply designs, as well as cooling subsystem limitations. Most of these constraints are due to cost or size considerations, but they can significantly limit the amount of power and cooling available in the system. Often the individual components, especially the processors, are capable of additional performance, but the power and thermal costs require the system to enforce limits to ensure safe, continued operation. The EnergyScale architecture provides the support necessary to do this dynamically rather than using static safety margins in the design. In addition, because of data-center design considerations, customers may need to impose even lower limits on power and heat than those of an individual system. Such a need may arise when the customer uses the system in a data center with power delivery or cooling limitations or when the organization needs or wants to limit power and heat for cost or environmental reasons.

Oversubscription protection

Many design points in the POWER6 product family offer some type of redundant power supply or delivery. The power supply system was originally intended to be fully redundant with the growth in power and heat, but this may no longer be so in every case. Thus, when one of the redundant power components fails or is removed, the other may not always be able to handle the full load. This situation is known as *oversubscription*. The EnergyScale architecture offers the regulation necessary to ensure continued operation during oversubscription situations without taking actions that have devastating performance consequences.

Performance extension

Sometimes, the system design constrains the POWER6 processor to a lower frequency than that which is desirable for a given workload. One of the goals of the EnergyScale implementation is to be able to extract the maximum performance from the system while ensuring that it operates within safe limits.

Power savings

At times, customers may want the system to operate efficiently, saving as much power as possible while limiting the amount of performance reduction. The EnergyScale architecture provides a power-saving mode in order to achieve balance between performance and power consumption.

Design principles

The EnergyScale architecture is based on design principles that are used not only in POWER6 processor-based machines but also in the IBM BladeCenter* and IBM System x* product lines. These principles are the result of fundamental research on system-level power management performed by the IBM Research Division, primarily in the Austin Research Laboratory.

Out-of-band management of power

System management can use two different communication and control paths: in-band and out-of-band. In-band management is performed using system processors and memory with code that runs inside or on top of the hypervisor and operating systems (OSs). Communication is over system network links that also carry application-related traffic. On the other hand, out-of-band management uses one or more service processors to execute the management logic, and it has a separate communication path to off-system management logic or, in some cases, back to management code running on the system itself. Although some functions of the EnergyScale architecture run in-band, the implementation is primarily an out-of-band power management scheme.

System-level power management

The EnergyScale implementation measures and manages the power of the entire central electronics complex (CEC) of the machine or, in some cases, the whole system board, not just processor power. Most power management schemes implemented to date have been exclusively or largely processor power management mechanisms. Although the processors are major contributors to system-level power, especially on smaller machines, they are not the only power consumers. An early study [1], for example, showed that on larger configurations, memory is an approximately equal contributor to total system power.

Measurement-based implementation

The EnergyScale implementation is measurement based, continuously taking measurements of voltage and current to calculate the power drawn. It uses temperature sensors to measure heat as well as performance counters to determine the characteristics of workloads. Unlike many other power management implementations, it does not attempt to project power and temperature, for instance, from system utilization.

Hard real-time measurement and control

When running out-of-band, the EnergyScale implementation relies on hard real-time measurement and

control to ensure that the system meets the specified power and temperature goals. Timings are honored down to the single-millisecond range. The control loop is based on standard principles of closed-loop feedback control and is subject to the standard forms of analysis that are used to ensure stability, accuracy, and reasonable settling times.

Multiple actuators

In order to control power at the system level, the EnergyScale implementation uses multiple actuators to alter the power consumption and heat dissipation of the processors and the memory in the system.

Guaranteed safety

The EnergyScale architecture contains a number of features that are designed to ensure safe, continued operation of the system during adverse power or thermal conditions as well as in certain cases in which the EnergyScale implementation itself fails.

Policy-driven power management

The EnergyScale architecture implements policies for power management that represent user-defined objectives and constraints imposed by the system design. The EnergyScale implementation includes user interfaces that display power and temperature data and allow the user to set and monitor policies. These user interfaces are extensions to preexisting ones, providing consistency with the other functions in the product line and other IBM server systems.

System implementations

Although the EnergyScale architecture is primarily a firmware implementation, it does have some hardware components. The support needed for it in the POWER6 chip is described elsewhere [2]. The machines built using the POWER6 processor cover a wide range in terms of system scale and cost, and the systems products use a number of distinct platform designs. Although the EnergyScale architecture is intended to be implemented for all POWER6 processor-based machines, only two reference system architectures—a rack-mounted machine and a server blade—are described in detail here.

Rack-mounted system

The rack-mounted, or tower server, reference design uses the POWER6 processor with up to four POWER6 processor chips, for a total of eight processor cores. It has a maximum memory size of 256 GB and space for up to 6 standard serial attached SCSI (small computer system interface) (SAS) or 12 small-form-factor SAS disks. The rack mount has a standard service processor, called the flexible support processor (FSP), running a Linux** OS-based code stack. Unfortunately, because of its implementation characteristics, it is infeasible to add a hard real-time component to the FSP code stack. In addition, for cost reasons, the EnergyScale implementation is optional for the rack-mounted reference design. Thus, the implementation uses an optional plug-in card, containing an H8-architecture microcontroller called the *thermal and power management device* (TPMD). The TPMD card also has a supplemental memory chip for data storage, a programmable oscillator, and interface logic for the connector. Communication with the FSP, the POWER6 chips, and the voltage regulator modules (VRMs) is via industry-standard I2C[1] interfaces. In addition, analog/digital (A/D) converters are used to collect component power and temperature data. The A/D converters collect power information for the entire board, each processor chip, the memory, and the disks, as well as one form of temperature data for the processors. For historical reasons, these A/D-based measurements are called *autonomic management of energy* (AME). The I2C interfaces are used to collect digital temperature sensor data and critical path information from the processors. **Figure 1** shows a diagram of a

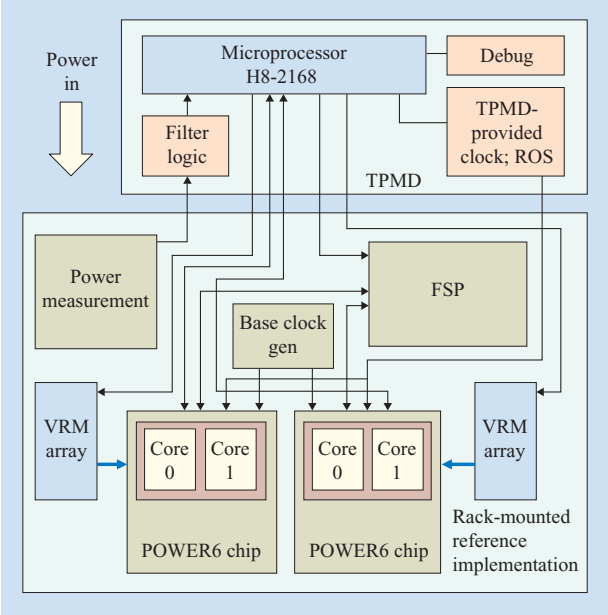

Figure 1

Diagram of a prototype HV8 board with only two POWER6 processor chips.

[1] *I2C* (inter-integrated circuit) is an industry-standard interface that is used to connect the chips on a system board to a service processor. Firmware uses it to collect information and exert low-level control of the connected components.

prototype of the rack-mounted reference design with two POWER6 processor chips.

There are two ways of controlling the power and heat of the processors. The first is to scale their voltages and frequencies. The design of the POWER6 processor-based machines is such that, in general, all of the processors in the system have to run at the same frequency. The TPMD card contains the oscillator that is used to clock the processors when the card is present and active, and the H8 can set the processor frequency by setting the oscillator frequency. On the other hand, each processor chip has a unique set of operating voltages that are associated with each frequency. The H8 uses the I2C interfaces to the VRMs to set them. Scaling frequency and voltage is a relatively slow operation that requires multiple steps to reach the target values, but it yields large power savings when scaling down since, in theory, power decreases as the product of the frequency and the square of the voltage. In practice, the savings are often less than what was anticipated, but they are still substantial with limited impact on performance. The second mechanism for managing processor power is called *processor throttling*, which delays instruction processing by injecting dead cycles. Although using processor throttling to reduce power is much faster, the power savings are approximately proportional to the amount of throttling, and the performance loss is also approximately linear. The EnergyScale implementation uses a combination of voltage and frequency scaling as well as throttling to implement processor power control.

The EnergyScale implementation manages the power of the system memory by using a combination of memory power-down and memory throttling. Memory power-down is a standard feature of the memory chips used in all POWER6 processor-based machines, and the embedded memory controllers on the POWER6 chips implement an intelligent strategy for using it. The memory chips still retain their contents, and it takes two memory clocks to power up any powered-down chips. In addition, the EnergyScale implementation can use the memory throttling feature of the POWER6 processor memory controller. Similar to processor throttling, memory throttling reduces the rate of memory accesses and, thus, the power consumed by the memory.

Following the typical conventions for all POWER6 processor-based system implementations, the FSP has an Ethernet interface (not shown in Figure 1) that provides a communication path between external programs and management consoles and the TPMD.

Server blade

Unlike the rack-mount design, the server blade plugs into a standard BladeCenter [3] chassis, which makes power management mandatory rather than optional. Rather than having a connector for an optional TPMD card, the blade has all of the hardware support for the EnergyScale architecture built directly into it. In addition, the FSP does not communicate by Ethernet directly with external programs and consoles. Instead, the FSP and the TPMD are controlled by the BladeCenter management module, and communication between the FSP and the management module is over a standard serial link using the RS-485[2] protocol. However, the EnergyScale implementation uses the same basic system features—processor frequency and voltage scaling, processor throttling, memory power-down control, and memory throttling—to regulate system power.

Calibration considerations

In order to determine which actions it should take, the power and thermal management logic in the EnergyScale implementation must have some basic information about the POWER6 processors. Each POWER6 chip has a unique set of power, thermal, and performance characteristics. To make this information available to the system, each chip contains module-level vital product data (VPD) such as the frequencies that it supports, their associated voltages, and the amount of power that the chip consumes at the specified operating point. This information is collected during manufacturing tests and is used to calibrate the behavior of the EnergyScale control system.

In addition, each system design has well-defined power supply and thermal cooling capacities. In some cases, there is redundant power, and the system has a lower power capacity when one of the redundant components is not operational. For example, with the server blade, the BladeCenter system has two power domains, each of which shares common power supplies and cooling. Each domain has redundant power supplies, and each domain is designed to operate with only a single supply operational. The blade has an interrupt that goes to the TPMD, notifying the TPMD when one of the redundant supplies has failed or has been removed. This supports the oversubscription management logic, which is described in a later section. The design of the BladeCenter system determines the oversubscription limit under which each blade must stay if the domain is operating using a single power supply.

Basic algorithms

The basic algorithms of the EnergyScale architecture are based on multiple closed-loop feedback controllers [4] that work together using a technique called *voting boxes*. This technique was originally developed to regulate power and temperature for the BladeCenter product line,

[2] *RS-485* provides an interface to connect data terminal equipment to data communication equipment.

and a recent paper [5] describes the techniques used in the development and calibration of the control loops. **Figure 2** shows the general form of the closed-loop system used by the EnergyScale architecture.

Each type of characteristic controlled by the EnergyScale architecture, including power, temperature, performance, and in some cases, the acoustical characteristics of the system, has its own control loop that takes the set-point and the sensor data collected by the TPMD, calculates the difference between the set-point and the measured value, and produces a control action intended to reduce any difference. With the EnergyScale implementation, the control actions are expressed as actuator settings such as throttling levels or frequency settings. Processor voltages are determined by frequency and are typically the lowest voltages that support the correct operation of the processor.

The EnergyScale implementation uses a number of logical sensors that are the firmware representation of the physical sensors being monitored. Sensors are then grouped by type: power, thermal, and performance. For power, the primary sensor used by the EnergyScale implementation is the board-level power as measured by AME. The thermal sensors are primarily based on the digital thermal sensors available on the POWER6 processor chip, but the EnergyScale implementation uses the on-chip wire resistors as calibration. All of the logical sensors are the result of firmware running on the TPMD using the raw data provided by the hardware and converting it into values that are then fed into the control loops. For example, the raw data for the AME sensor include separate voltage and current readings that are filtered and then used to calculate power. To measure performance, the EnergyScale implementation uses a millions-of-instructions-per-second (MIPS) sensor, which is calculated in the firmware by multiplying the currently measured instructions per cycle (IPC) by the frequency of the processors.

Each control loop receives the current values of the sensor information and its set-point as input. It uses a proportional controller to calculate the control action for the actuator. Each loop may have its own period since some sensors change values more slowly than others. The EnergyScale implementation takes the current output of each control loop and passes it into a voting box. In the simple case in which only a single actuator is used uniformly, such as in throttling, there is one voting box. The voting box logic compares its inputs and selects the most restrictive one; the idea is that the most restrictive one represents the control loop with the most stringent, or lowest power, requirement. For an actuator that throttles multiple processor cores and the memory, the throttling level is always uniform for all of them. Different implementations of the EnergyScale firmware use different actuators, with earlier ones using throttling and later ones using voltage and frequency scaling. However, the design allows for the use of multiple actuators, in which case, there is one voting box per actuator, and the outputs of the control loops are fed into all of the voting boxes.

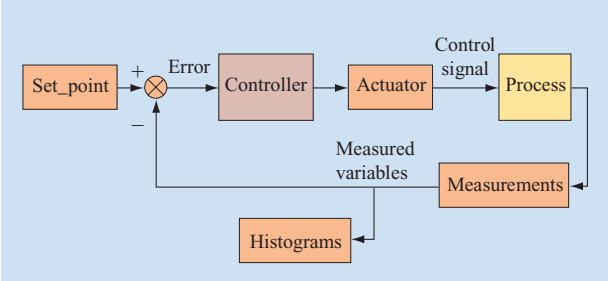

Figure 2

General structure of the closed-loop control used by EnergyScale.

Oversubscription is handled slightly differently since it is considered an emergency, one that must be handled immediately. When the TPMD receives an oversubscription interrupt, it overrides the current set-point and uses the control mechanism to drive the power under the oversubscription limit. The power reduction is potentially quite substantial, from about 430 W to 370 W, for the blade reference design. The EnergyScale implementation uses throttling to get under the oversubscription limit initially but may, in some implementations, reduce the frequency and voltages over time while relaxing the throttling level in order to minimize the performance impact.

In addition to managing power, temperature, and performance, the EnergyScale implementation also collects data for trending purposes. It places data into histograms, with separate histograms for power, temperature, and performance. Each histogram divides the range of values for its sensor into buckets and, for each bucket, has a count of the number of samples measured to be within the bounds of the bucket. In addition, the EnergyScale implementation collects two kinds of running counts that are expressed as 8-byte registers. The first is called the *energy accumulation register* (EAR) that has a running count of the number of joules consumed. The second register is called the *clock accumulation register* (CAR), which provides the average frequency of the processors, taking into account the effects of frequency changes and throttling. Actually, there are also multiple CARs, reflecting the effective frequency over a 1-minute, a 30-minute, and a programmable duration.

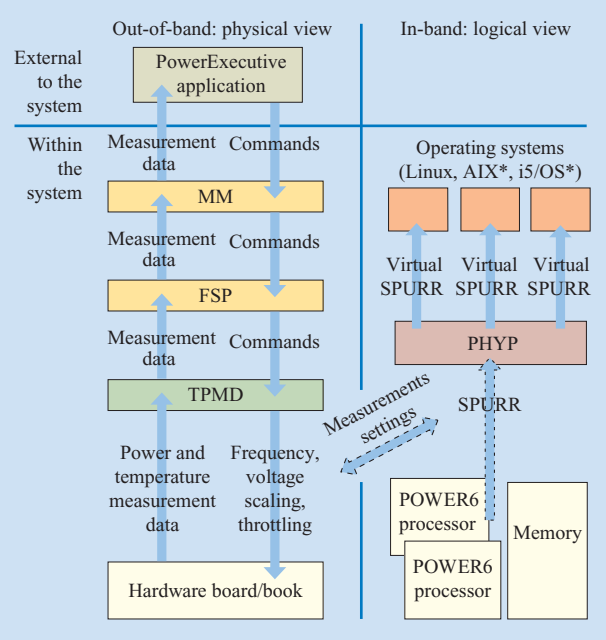

Figure 3

The structure of out-of-band and in-band management in a POWER6 processor-based system.

Management policies

The EnergyScale design described in this paper supports a number of power and thermal management policies, although not all of them are provided with every release or on every system. The policies include the following.

- *Benchmark turbo*—Benchmark turbo maximizes the single-threaded performance of the system by putting one core of each processor chip into a lower-power state such as the POWER6 processor nap mode and by using the extra power and cooling capacity in support of the remaining core; doing so provides the capability to increase the frequency and voltages of the processors to values higher than their nominal ones. To further maximize single-threaded performance, the OS can turn off simultaneous multithreading (SMT) in order to put each running core into a completely single-threaded state.
- *Maximum performance*—Under the maximum performance policy, the EnergyScale architecture regulates the system in such a way as to attempt to extract the maximum performance for the workload by using as much power as possible without violating the power or thermal limits of the box.
- *Power cap*—With the power cap policy, the EnergyScale implementation enforces a specified power cap (maximum). The user or, for the server blade, the management module may set, change, or remove the power cap at any time.
- *Fixed performance*—On systems other than the server blade, the user may specify that the EnergyScale firmware simply collect power and thermal data but not take any power management actions. Unlike the other policies, this one guarantees fixed performance at the nominal frequency of the machine. This policy cannot be used on the blade.
- *Maximum power savings*—Under this policy, the EnergyScale implementation attempts to save as much power as possible for a given workload.
- *Optimal power/performance*—With this policy, the EnergyScale implementation utilizes heuristics and algorithms to pick the most optimal power-versus-performance trade-off on the basis of workload characteristics and the power and thermal environment. The user may specify a bound on the performance lost, also known as a *performance floor*. If the system approaches the performance floor, the EnergyScale implementation increases power management settings in order to avoid going below the floor value.

These policies are set by the user or administrator of the system using the firmware and software described in the next section.

Firmware and systems software implementation

Except for the platform hardware discussed previously, the EnergyScale architecture is implemented by firmware running on the TPMD and the FSP of the system and the IBM PowerExecutive* management software, which executes as a plug-in to the IBM Director software. Additionally, although the EnergyScale implementation is primarily an out-of-band power management design, managing system-level power and temperature has some effects that are visible to in-band software. In particular, power management results in performance variability. *Performance variability* refers to the fact that as the power management implementation operates, it can change the effective speed of the processor. **Figure 3** shows the relationship between the components of the EnergyScale firmware and the software implementation, including the management module, which is used only for blades. The structure for the rack-mounted design is similar but does not include a management module.

TPMD firmware

The thermal and power management firmware (TPMF) runs on the TPMD and in hard real time implements the algorithms described previously. Hard real-time

environments guarantee timings to a specified precision, and if a time-critical operation fails to complete within its specified period, it is treated as an error. Hard real-time environments are commonly used for control applications; the role provided by TPMF is that of a control function. However, TPMF also includes additional functions for communicating with the FSP and for managing itself. The real-time control runs as an interrupt handler driven by a timer interrupt, which occurs every millisecond. Most of the remaining functions execute in an environment provided by the Express Logic ThreadX**,[3] real-time OS. The two parts of TPMF communicate with each other through shared data structures.

TPMF operates under the control of the firmware that runs on the FSP. Since the TPMD operates only as an I2C slave to the FSP, all communication is in the form of commands sent by the FSP and is processed by a command handler running on the ThreadX OS. In many cases, the commands are polls for data, and the TPMF responds by sending back power and thermal measurements that it has collected and stored in the TPMD memory. There are cases in which TPMF must indicate errors or other important conditions to the FSP. The hardware provides a general-purpose input/output (GPIO) line that the TPMF can raise, which interrupts the FSP and causes it to poll. Since the FSP is not real time in any sense, the interval between the request for a command and its arrival is unpredictable, and commands may not arrive in the order in which the TPMF expects. The implementation contains buffering logic and a sequencing protocol to allow TPMF to respond correctly.

In addition to monitoring the power, temperature, and performance of the system, TPMF also monitors its own health. The health monitor periodically checks the hardware sensors for errors and ensures that the rest of the control software is operating properly.

Service processor firmware

Prior to the development of the EnergyScale implementation, firmware running on the FSP set up the processor frequency and voltages and performed all of the health monitoring of the system. With the introduction of the EnergyScale implementation, the role of the FSP firmware is somewhat different. For the EnergyScale architecture, the FSP is responsible for directly controlling the TPMD as well as for providing a communication path both to the TPMD for policy information and settings and back from it for power, thermal, and performance data. The FSP acts as the master on the I2C interface to the TPMD, issuing periodic commands to collect data and in response to interrupts from it. The FSP initializes the TPMD, and in later implementations that use frequency and voltage scaling as an actuator, the FSP transfers control of the TPMD oscillator and the VRMs to TPMF. Moreover, the FSP is responsible for updating TPMF as needed.

In the implementations other than that for the server blade, the system can enter safe mode if the TPMD fails. *Safe mode* is a power and thermal management state that is known to support continuous operation without overheating the machine or exceeding the limits of its power delivery system. There are well-defined voltage and frequency settings for it, and these may, in fact, be higher than the ones being enforced by the TPMD if there is, for example, a power cap. For the blade, the BladeCenter system has safety logic that shuts off the blade when a TPMD failure causes an overpower or overtemperature situation.

User interface and controls

The PowerExecutive [6] software is the IBM strategic interface for systems-level power management. It is already in use with BladeCenter and certain System x machines. The PowerExecutive interface shows trending information for power, temperature, and system speed, and it allows the user to specify a power and thermal management policy and its parameters. For consistency, the EnergyScale architecture also uses the PowerExecutive interface so that a single instance of the PowerExecutive software can manage a heterogeneous set of systems.

The rack-mounted reference design uses the PowerExecutive interface supplemented with a new communications interface using the Common Information Model (CIM) [7]. The FSP uses the CIM to package the power, temperature, and performance information that it collects, and the PowerExecutive software sends commands and policy information to the FSP using the CIM. In this instance, the EnergyScale architecture uses the CIM as a transport wrapper rather than implementing a full model of power management.

Since the blade is a BladeCenter blade and must conform to the BladeCenter power and thermal management architecture, the power, thermal, and performance information flowing out of the FSP goes to the management module. The management module collects all the information for all the blades in the BladeCenter system, which may include other types of blades as well as the POWER6 processor-based blades, and buffers it for the PowerExecutive interface. In addition, the management module performs power budgeting by collecting the current power consumption level from each blade and assigning a power budget to it.

This allows the management module to control the overall consumption of the BladeCenter system to ensure safe operation and to meet any cap on the power of the

[3] *ThreadX* is a very small, real-time operating system for microcontrollers and embedded devices developed and marketed by Express Logic, Incorporated.

whole BladeCenter system imposed by the PowerExecutive interface.

System software impacts

As indicated previously, the EnergyScale implementation has an impact on the system software, including the hypervisor and all three OSs supported by System p* and System i*—Linux, AIX, and i5/OS—by introducing performance variability. In addition, there are three other areas in which the EnergyScale implementation forces changes or additions.

- Idle power reduction. Historically, Power Architecture technology-based machines consumed nearly their maximum power when idle. This is no longer acceptable to customers or regulators. The EnergyScale implementation reduces power consumption when the machine is idle by using a lower power mode of the processor.
- Accurate idle detection. To avoid making incorrect power allocation decisions among the processor cores in the system, the EnergyScale firmware must be able to accurately determine how idle a hardware thread is. Doing so requires support in the OSs.
- Capacity upgrade on demand. Capacity upgrade on demand (CUoD) is a feature of the Power Architecture technology-based machines that allows a customer to purchase a machine with a certain number of processors but license only some of them. As the customer requires additional capacity, additional licenses can be purchased. The EnergyScale firmware affects both the handling of unlicensed processors and the decision-making process for purchasing additional capacity.

Performance variability

The POWER5* processor family introduced SMT. The natural way for system software to use SMT is to assign a logical processor to each thread, so the OSs use two logical CPUs (central processing units) per processor core. However, this alters the time accounting and utilization calculations; the amount of time in an interval is no longer the difference in the number of ticks recorded at the beginning and at the end of an interval of time since the cycles represented by the ticks are shared between two logical processors. To support accurate accounting and utilization, the POWER5 architecture added a special-purpose register for each SMT thread, called the *processor utilization of resources register* (PURR). The PURR counts only the timebase ticks assigned to the thread by the processor. The hypervisor virtualizes the PURR. For dedicated processor partitions, in which processor cores are dedicated to running code only for that partition, the virtualized value is simply the total PURR count for the hardware thread. On the other hand, for processor cores in the shared processor pool, the hypervisor saves and restores the PURR value, managing a separate PURR value for each partition sharing the processor core.

For accounting purposes, the amount of CPU time consumed in an interval is the value of the virtualized PURR at the end of an interval minus the value of the virtualized PURR at the beginning. The utilization of the logical CPU is the ratio of the number of PURR ticks spent in the active state, that is, outside of the idle state, to the total number of PURR ticks for the interval. However, the utilization of the physical CPU is the sum of the PURR ticks in the active state for the two threads divided by the number of timebase ticks in the interval.

In order to provide the same level of support to the OSs in a power-managed environment, the POWER6 processor still has a PURR for each SMT thread, but it also contains an additional special-purpose register for each hardware thread, called the *scaled processor utilization of resources register* (SPURR). The SPURR is used to compensate for the effects of performance variability on the OSs; the hypervisor virtualizes the SPURR for each hardware thread so that each OS obtains accurate readings that reflect only the portion of the SPURR count that is associated with its partition. The implementation of virtualization for the SPURR is the same as that for the PURR.

Building on the functions provided by the hypervisor, the OSs use the SPURR to do the same type of accurate accounting that is available on the POWER5 processor-based machines. With the introduction of the EnergyScale architecture for the POWER6 processor-based machines, not all timebase ticks have the same computational value; some of them represent more-usable processor cycles than others. The SPURR provides a scaled count of the number of timebase ticks assigned to a hardware thread, in which the scaling reflects the speed of the processor, taking into account frequency changes and throttling, relative to its nominal speed. TPMF determines the scaling based on the frequency and throttling values that it sets and uses its interface to the processor chips to set the appropriate control values for the SPURR. The result is that the ratio of the SPURR over a time interval to that of the PURR over the same interval reflects the scaling of the processor and the performance effects of power management on it. The OSs use the virtualized SPURR to calculate exact accounting information for the processes that they run: The change in the SPURR value in an interval is a scaled number of timebase ticks that the hardware thread ran. On the other hand, processor utilization is more complex in a power-managed environment. Traditional CPU

utilization continues to be calculated using the PURR, because processor utilization is expected to increase as the processor is scaled down and decrease as the processor is scaled up. However, there is another metric that has some value: the ratio of the SPURR spent in the active state to the total SPURR value of the thread. This value reflects the load on the logical processor independent of scaling, and if the EnergyScale firmware can alter the power management settings (e.g., when the policy is meant to only save power), a SPURR-based calculation suggests that the processor can do much more work as the power management settings increase.

Performance variability also affects some of the performance tools and metrics built with the user-visible performance counters (not counters dedicated to the EnergyScale firmware, which are separate). Many of these counters count processor cycles, and since the number of cycles per unit time is variable, the values reported by unmodified performance monitors are subject to some interpretation. In addition, if the EnergyScale firmware changes the effective speed of the processor by altering the throttle settings or the frequency during a data collection interval, the rate of cycle counting changes. One good example of the type of interpretation and possible modification needed is in the collection of a traditionally important metric, IPC (which is often used to indicate how efficiently a particular program runs on a given piece of hardware), and the focus of many optimization efforts, both hardware and software, is on increasing the IPC of high-value programs. If the program runs on a POWER6 processor-based system with a power cap that is being enforced by the EnergyScale firmware by voltage and frequency scaling, then because the frequency is lower than the nominal value, the cycle count collected from the performance counter is lower for the same sampling interval length than it would be if the system were running at nominal frequency. If the program makes heavy use of memory and the goal is to understand the behavior of the program on the hardware, the IPC reported is higher than the intuitively correct value obtained from performance of the same test at the nominal frequency. Whether this provides the correct information depends on the goal of the analysis. If the goal is to understand the behavior of the system as configured, the higher IPC value correctly reveals that a memory-bound program is affected less by scaling the frequency down.

Idle power reduction

A major problem with earlier Power Architecture technology-based machines is that their power consumption is very high when they are idle, because they run a tight, predictable processor-bound idle loop. The POWER6 processor adds a new low-power mode called *nap* that stops processor execution. When both hardware threads of a processor core enter nap mode, the whole core enters nap, which allows the hardware to turn off many of the circuits inside the core, reducing power consumption and allowing the temperature to decrease, further reducing power. Exiting nap mode is sufficiently rapid for its use in many circumstances as an idle state. When an OS yields control of a hardware thread to the hypervisor, the hypervisor determines whether the processor core is in a shared processor pool. If so, and if there is no other partition to dispatch, the hypervisor puts the thread into the nap state. If the core is in the dedicated processor pool, then it always puts the thread into nap mode.

Accurate idle detection

In order to be performance aware, the EnergyScale implementation uses the POWER6 processor hardware facilities to read dedicated performance counters out-of-band. However, it is possible to fool the performance control loop in the TPMD if the OSs run their traditional idle loop for an extended period since, as noted previously, the idle loop is a very computationally intensive piece of code. Although the nap mode is easily detected as an idle state using the dedicated performance counters, the OSs do not necessarily put idle processors into nap mode immediately. There can be an indefinite delay, based on user-controlled policy settings. Thus, both hardware threads may run the idle loop for extended periods and appear to be performing processor-intensive, useful work. If there is no other notification of the idle state, the out-of-band code interprets this phenomenon as a CPU-bound program that should run at a high frequency with little or no throttling. This wastes power and can even cause the EnergyScale firmware to shift power from another processor core running a memory-bound but useful application. To avoid wasting power or unnecessarily penalizing other programs, the OS schedulers set another special-purpose, per-hardware-thread register, the run latch, when they dispatch useful work and reset it when they enter the idle state, whether they are running the idle loop or giving control back to the hypervisor to enter nap mode. TPMF reads a counter in the POWER6 core that reflects the state of the run latch and counts only those cycles that occur when the run latch is set, not those spent in the idle or nap state. This counter allows TPMF to recognize accurately whether the hardware thread is running useful work, and it ensures that it does not mistake idle for an important, computationally bound program. If both threads of a core are idle a significant fraction of the time, it becomes a target for throttling, if necessary. In addition, if all cores are idle a significant fraction of the time and the policy is to save power, TPMF can scale down frequency and the

processor voltages in order to reduce the overall power consumption.

Capacity upgrade on demand

POWER6 processor-based systems support CUoD, a feature that allows the customer to purchase a machine but pay for a license to use only some of the processors. This allows the purchaser not only to save money on the initial acquisition but also to be able to increase the machine capacity by purchasing additional processor licenses at a later time. On previous-generation machines, the unlicensed cores ran the idle loop and consumed almost maximum power, creating regulatory problems in some countries. With the POWER6 processor-based systems, the hypervisor holds the unlicensed cores in nap mode to minimize their overhead.

In some cases CUoD may yield unexpected results. Since unlicensed cores consume less power than active cores, the additional power that they consume when licensed may cause the EnergyScale firmware to reduce processor frequency, for example, in order to stay under a power cap. In extreme cases, licensing an additional processor core could actually reduce the number of available processor cycles. However, the trending information available with the EnergyScale implementation allows the user to determine the effect of licensing additional cores by showing how much additional power the machine can consume before taking a power management action. This, combined with information about the expected additional power consumption due to the proposed licensing action, allows the customer to make a rational decision about when to license additional processor cores.

Extensions to other system designs

Although the basic design of the EnergyScale architecture is similar for all of the POWER6 processor-based systems, there are two special cases. The first is for certain POWER6 processor-based midrange products, which are based on a POWER5 processor system design. These systems do not have onboard TPMDs and they do not have the connectors for optional TPMD cards. However, because of the pressing customer need for power measurement and management, these systems offer two power management-related functions. The first is a form of power measurement using intelligent power distribution units (IPDUs). These devices contain power meters on their power outlets, and new support in the PowerExecutive software allows it to collect measurement data from them. With the line cords of the machine plugged into an IPDU, its readings can be used to monitor the power consumption of the system to measure trends. This requires the user to manually enter the correct configuration data to associate the system with the plugs that it is using. The second power management feature is a simple, static power-saving mode implemented using the FSP. As noted previously, prior to the introduction of the TPMD, the FSP configured the frequency and voltages of the processors. The midrange implementation of the static power-saving mode uses the FSP to put the processors into a predefined lower frequency and voltage state. Control is manual through an interface provided by the FSP or through the PowerExecutive software, so a typical use for the feature is to reduce power consumption when the workload is predictably low such as during the overnight hours.

The second special case is for the largest POWER6 processor-based machines. These machines contain multiple boards, and the designs of their predecessor in the POWER5 product line already contain some power measurement features. Such machines pose a significant challenge because of their scale and the additional complexity imposed by their hardware designs. The design approach for extending the EnergyScale architecture to them involves three changes. First, the EnergyScale architecture uses the existing power measurement function provided by the bulk power controllers (BPCs) used in the power supplies. Second, rather than adding a TPMD to each board, the design uses existing microcontrollers that are already embedded in the power distribution subsystem. This allows real-time control on each board. Third, system-wide changes such as changes to the frequency as well as the reporting of system-wide measurements use non-real-time implementations running on an FSP. Although this limits the responsiveness of the power management system, it allows it to scale to the scope needed to control a very large machine.

Concluding remarks

A demand exists for systems that track their power consumption and provide features that allow management of their power and temperature. The EnergyScale architecture provides the system-level power and thermal management support that is needed to make the POWER6 processor-based products meet these important requirements. The EnergyScale architecture is based on the results of research work conducted by the IBM Research Division over the past 5 years, but its implementation is the result of an intense collaboration between the Research Division and the Systems and Technology Group (STG). It is based on the fundamental principles of measurement, control, and reporting. POWER6 processor-based systems are heavily instrumented at the microprocessor and board levels, allowing firmware to collect power, temperature, and performance data in real time without interfering with the workload running on the system. The data collection code

runs in real time on a dedicated microcontroller and uses the information for closed-loop, feedback control of the system to ensure that the set power, temperature, and performance goals are met. The microcontroller has a number of actuators that it uses to manage power and temperature and ensure that the system operates at the specified set-points. The measured data is also sent to higher-level firmware and software that provide the user with trend data. User interface logic converts policies to precise directives to the control system.

Although the power management system operates out-of-band, its behavior has some effects on the workloads running on the system. In particular, it causes the system to run at different speeds at different times. This, in turn, requires changes in the accounting and performance management logic in the OSs as well as some basic support in the hypervisor.

Not all of the features described here will be available with the initial shipment of the POWER6 processor-based products or on all machines in the product family. Instead, the features will be delivered in phases in order to control development expense and to ensure product quality. However, most machines will require only a firmware upgrade to enable additional EnergyScale architecture features as they are released.

The EnergyScale implementation on POWER6 processor-based machines is simply an initial implementation of power management for the IBM System p and System i machines. The next generation of processors and systems will apply the same basic principles to managing power, temperature, and performance, but they will also have capabilities that will allow finer control mechanisms on shorter time scales in order to offer more performance at lower power and temperature.

Acknowledgments

Andreas Bieswanger, Nick Bofferding, Soraya Ghiasi, Mike Hollinger, Tom Keller, Naresh Nayar, Karthick Rajamani, Guillermo Silva, Randy Swanberg, and Christine Wang contributed to the design and implementation. Members of the PowerExecutive team, including Tom Brey, Scott Piper, Cale Rath, Mike Turner, and Jeff VanHeuklon, helped with the architecture and the implementation of PowerExecutive support for the EnergyScale architecture. Josh Friedrich, Hal Chase, and Norm James contributed to the design and characterization of the power management features of the POWER6 chip. Brad McCredie is the technical sponsor of the project. Lorraine Herger committed the resources of the Austin Research Laboratory to getting it done, and the entire STG organization has supported its design and implementation.

References

1. C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller, "Energy Management for Commercial Servers," *IEEE Computer* **36**, No. 12, 39–48 (2003).
2. M. S. Floyd, S. Ghiasi, T. W. Keller, K. Rajamani, F. L. Rawson, J. C. Rubio, and M. S. Ware, "System Power Management Support in the IBM POWER6 Microprocessor," *IBM J. Res. & Dev.* **51**, No. 6, 733–746 (2007, this issue).
3. D. M. Desai, T. M. Bradicich, D. Champion, W. G. Holland, and B. M. Kreuz, "BladeCenter System Overview," *IBM J. Res. & Dev.* **49**, No. 6, 809–822 (2005).
4. J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*, Wiley-Interscience, Hoboken, NJ, 2004.
5. C. Lefurgy, X. Wang, and M. Ware, "Server-Level Power Control," *Proceedings of the Fourth International Conference on Autonomic Computing*, 2007; see *http://www.ece.utk.edu/~xwang/papers/icac_power.pdf*.
6. IBM, PowerExecutive; see *http://www.ibm.com/systems/management/director/extensions/powerexec.html*.
7. Common Information Model; see *http://www.dmtf.org/standards/cim*.

Received January 9, 2007; accepted for publication July 30, 2007; Internet publication October 23, 2007

Hye-Young McCreary *Global Firmware Development, IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (hmc@us.ibm.com)*. Ms. McCreary is a Distinguished Engineer and is responsible for delivering Power Management Technology Solutions for POWER6 processor-based systems. She joined IBM in 1988 as a UNIX** file system expert working on the first release of AIX for the IBM RS/6000* platform. She worked on many key AIX technology projects, focusing on AIX scalability, performance, and quality. Having grown through a wide variety of technically challenging projects and assignments in AIX development, she moved to Global Firmware Development in 2003, where she is responsible for delivering the IBM on-demand, virtualization, and reliability, availability, and serviceability (RAS) solutions for System p and System i servers. Ms. McCreary is also a Chairman of the Platform System Design Board for System p, System i, and storage systems.

Martha A. Broyles *Global Firmware Development, IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (mbroyles@us.ibm.com)*. Ms. Broyles is a Software Development Engineer in the Systems and Technology Group at IBM. She joined IBM in 1999 as a member of the Systems Power Control Network team for IBM servers. In 2005 she became a member of the EnergyScale team, where she has been working on the service processor firmware design to provide more energy efficient servers. Ms. Broyles received a B.S. degree in mathematics with honors from the University of Minnesota Duluth.

Michael S. Floyd *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (mfloyd@us.ibm.com)*. Mr. Floyd is a Senior Engineer specializing in RAS and power-efficient design of server microprocessors and systems. He received a B.S. degree in computer engineering from the Georgia Institute of Technology in 1995 and an M.S. degree in electrical engineering from Stanford University in 2000. His 12 years of experience with IBM include bring up, test, and debug of the Power PC 620* and POWER4* microprocessors, in addition to holding RAS design, lead, and microarchitecture definition roles for the POWER4, POWER5, and POWER6 microprocessors and support chips. After leading the POWER6 microprocessor RAS design, he worked with IBM Research Division to develop the POWER6 system power management implementation. Mr. Floyd, who is currently the POWER7* Adaptive Power Management Design Leader/Architect, holds 27 patents and has 47 more pending. He has been named an IBM Master Inventor and has co-authored four published papers.

Andrew J. Geissler *Global Firmware Development, IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758*. Mr. Geissler is an Advisory Software Engineer who leads a team working on power management of IBM servers. Prior to this, he worked on service processor firmware for System i and System p servers. His main focus at IBM has been working on embedded microprocessors to initialize and control the IBM Power Architecture technology-based systems. Mr. Geissler received a B.S. degree in computer engineering from the University of Maine in 2001.

Steven P. Hartman *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (sph@us.ibm.com)*. Mr. Hartman is a System Design Architect serving as the Chief Engineer for the POWER6 Blade program. He has been responsible for system design of the PowerPC 603*, Power PC 620, POWER4, POWER5, and POWER6 processors and for system implementations ranging from very large System p machines to the midrange and low-end models. During his 30 years at IBM, he has been responsible for the development of optical scanning and character recognition systems, RS/6000 workstations, embedded microprocessor subsystems, and dynamic power thermal control subsystems.

Freeman L. Rawson *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (frawson@us.ibm.com)*. Mr. Rawson is a Senior Technical Staff Member in the Power-aware Systems organization in the Austin Research Laboratory. Prior to coming to the Research Division, he spent 23 years in IBM development working on a wide variety of systems-related projects. His research interests include applied artificial intelligence, machine learning, systems management, operating systems, and systems architecture. He received a B.S. degree in mathematics from Michigan State University and a Ph.D. degree in philosophy from Stanford University. He holds 20 patents, has 20 more patent applications pending, and has published more than 20 refereed technical papers. He is a member of the IEEE Computer Society, the ACM, and AAAI.

Todd J. Rosedahl *Global Firmware Development, IBM Systems and Technology Group, 3700 Highway 52 N., Rochester, Minnesota 55901 (rosedahl@us.ibm.com)*. Mr. Rosedahl is a Firmware Architect on the service processor for System x, System i, and System p servers. He has been responsible for the power control I/O network and is currently the pervasive function owner for the power and thermal management firmware. He has worked for IBM for 15 years and has numerous patents.

Juan C. Rubio *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (rubioj@us.ibm.com)*. Dr. Rubio is a Research Staff Member at the IBM Austin Research Laboratory. He received a B.S. degree in electrical engineering from Universidad Santa Maria La Antigua, Panama, in 1997 and a Ph.D. degree in computer engineering from the University of Texas at Austin in 2004. His dissertation explored a hierarchical architecture to improve data access in large enterprise workloads. At IBM, he has studied architectural techniques to monitor, model, and manage power and temperature in servers.

Malcolm S. Ware *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (mware@us.ibm.com)*. Mr. Ware received a B.S. degree in electrical engineering from Purdue University in 1983 and an M.S. degree in computer architecture and communications from North Carolina State University in 1986. He spent his first 10 years with IBM at the Research Triangle Park (RTP) facility developing speech and image coding algorithms, music synthesizers, and low-speed modems for the Mwave DSP. In 1993 he went on international assignment for 5 years to the IBM Zurich Research Laboratory (ZRL) in Switzerland, and he worked with IBM Fellow Gottfried Ungerboeck on high-speed modems including V.34 and V.90 for the Mwave products shipped in Lenovo ThinkPad** laptop computers. After returning to RTP for 2 years to examine broadband and network processing opportunities, he spent 16 months on assignment again at ZRL developing prototypes of Asymmetric Digital Subscriber Line (ADSL), Symmetric High-speed Digital Subscriber Line (SHDSL), and Very high-speed Digital Subscriber Line (VDSL) broadband transceivers. From 2002 to 2003, Mr. Ware studied wireless transmission systems at RTP and then power modeling for embedded systems. In 2003 he joined IBM Austin Research Laboratory to pursue power-aware systems including the IBM server-class systems for both System x and System p, with a primary focus on closed-loop control systems to manage power, thermals, and performance dynamically. He was promoted to Senior Technical Staff Member in 2006, and he holds 39 patents.