IBM Intelligent Bricks project—Petabytes and beyond

W. W. Wilcke
R. B. Garner
C. Fleiner
R. F. Freitas
R. A. Golding
J. S. Glider
D. R. Kenchammana-Hosekote
J. L. Hafner
K. M. Mohiuddin
KK Rao
R. A. Becker-Szendy
T. M. Wong
O. A. Zaki
M. Hernandez
K. R. Fernandez
H. Huels
H. Lenk
K. Smolin
M. Ries
C. Goettert
T. Picunko
B. J. Rubin
H. Kahn
T. Loo

This paper provides an overview of the Intelligent Bricks project in progress at IBM Research. It describes common problems faced by data center operators and proposes a comprehensive solution based on brick architectures. Bricks are hardware building blocks. Because of certain properties, defined here, scalable and reliable systems can be built with collections of identical bricks. An important feature is that brick-based systems must survive the failure of any brick without requiring human intervention, as long as most bricks are operational. This simplifies system management and allows very dense and very scalable systems to be built. A prototype storage server in the form of a $3 \times 3 \times 3$ array of bricks, capable of storing 26 TB, is operational at the IBM Almaden Research Center. It successfully demonstrates the concepts of the Intelligent Bricks architecture. The paper describes this implementation of brick architectures based on newly developed communication and cooling technologies, the software developed, and techniques for building very reliable systems from low-cost bricks, and it discusses the performance and the future of intelligent brick systems.

Motivation

The Intelligent Bricks project addresses four common data center problems that surfaced consistently during many discussions[1] between project team members and customers. From customers, we learned that

- *Today's systems are too difficult to manage*. System management expenses dominate the total cost of ownership [1, 2].
- *Systems should be more scalable and of lower cost*. The three independent scaling parameters are entry cost, granularity of scaling, and scaling range. The capital cost of systems today is compared against that of racks of low-cost, generic personal computers connected by Ethernet.
- *Better environmental parameters are needed*. For some customers, environmental parameters—system floor space, power, cooling, and noise—are the most important ones.
- *Better reliability and availability are desirable, but not at any price*. The best systems should be much more reliable and available than any system now available. However, while data loss events and outages are extraordinarily expensive for some applications, they are minor issues for others. Systems should allow operators to choose their own reliability and cost tradeoffs.

[1]Between 2001 and 2005, these data center issues and the ideas described here were discussed during some 70 customer visits to the IBM Almaden Research Center.

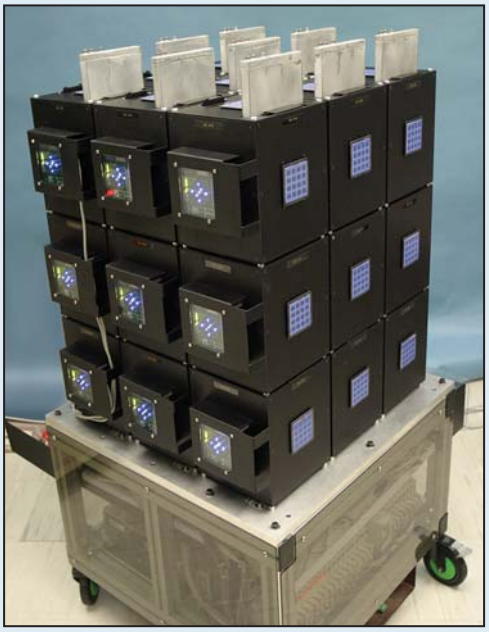
Brick-based system architectures

At first glimpse, the four problem groups appear to be completely unrelated—surely there is no relation between simplifying management and reducing floor space. However, it turns out that there is a common approach—brick architecture—that addresses all four problems simultaneously.

Zen of simplicity

The guiding principle of the Intelligent Bricks project is the pursuit of simplicity. Simple systems are easier to explain, easier to build, easier to maintain, and easier to

0018-8646/06/$5.00 © 2006 IBM



(a)

(b)

Figure 1

(a) IceCube: an operational intelligent brick storage server. (b) Internal view of one prototype brick.

use. They reduce errors in design, in production, in management, and in operation. To make a system simple, we make it very modular; a system of virtually any size can be built with identical bricks that encapsulate complex components but expose only well-defined interfaces. This simple approach judiciously reduces customer choices, freeing them from the complexity that arises when a system grows to become an unmanageable hodgepodge of multiple components, subsystems, and interconnections.

The Intelligent Bricks project has created a prototype, called IceCube, of such a modular, brick-based system. **Figure 1** shows the system, which is a cube of $3 \times 3 \times 3$ bricks, and the internal details of a single brick.

Definition of a brick-based architecture

Our principle of simplicity leads to what we call a *brick-based* system architecture. There are eight criteria, most of which should be fulfilled for an architecture to be classified as brick-based:

1. Hardware is encapsulated into physical units called bricks. Systems are collections of bricks.[2]
2. Bricks are indivisible units of system design and may not be modified or repaired in the field. Conceptually speaking, bricks are welded shut.
3. All bricks must contain processing, networking, and storage functionality. The latter may be minimal in a brick used as a compute engine and very large for storage-oriented bricks.[3]
4. Only a small number of different types of bricks should be needed to build an entire system.
5. Bricks must have long-lived and public interfaces for software and for the interfaces defining form factor, power, cooling, and networking.
6. Bricks that are directly interoperable are said to be members of a *brick family*.
7. Bricks should use low-cost, commodity components and should minimize internal redundancy.
8. Any brick in a system may fail, and the system must continue to operate without noticeable impact as long as most of the bricks in the system are alive. This property, called *fail-in-place*, requires system-level redundancy and allows maintenance to be deferred or even eliminated.

These criteria are inspired by a biological analogy; bricks correspond to cells[4] and systems to multicellular organisms. Any given cell in an organism can die without detectable consequences; the same should hold for sufficiently large brick-based systems.

Brick architectures and the problem set

This section provides a high-level summary of the way in which brick architectures address the four data center problems. The remainder of the paper elaborates on these arguments.

[2]Bricks are not required to have the physical shape commonly associated with the word. For example, a system using blades or 1U servers qualifies as brick-based if the other criteria are fulfilled. (See the next section.)
[3]Bricks should not be confused with thin clients managed by a central server.
[4]There is no relation to the IBM Cell Broadband Engine** processor.

- *Simplified management:* In a brick-based system, there is no need to deal with failed components quickly, if ever. Physical maintenance is limited primarily to adding new bricks. The system is easy to configure because there are only a few parts to deal with. Some implementations (for example, the prototype), contain no internal cables, fans, or other failure-prone items. With the proper software, many routine configuration tasks can be performed automatically.
- *Scalability and lower cost:* Scalability is inherent in a distributed, modular system. In some implementations, as demonstrated by the prototype, very high communications bandwidth and dense physical packaging allow scaling to thousands of bricks. Brick systems can be low-cost because bricks may use commodity components. Expensive central switches are eliminated. A cost disadvantage specific to storage servers is the fact that there is more processing power per disk in a brick system than in a conventional system. This is a tradeoff between hardware cost and reducing the (high) cost of system management.
- *Better environmental parameters:* Brick-based three-dimensional (3D) systems can be built at extremely high density. If liquid cooling is employed, thermal management of data centers is simplified, and floor space is reduced even further because systems can be placed close to one another. Fan noise is eliminated. Liquid cooling also reduces overall power consumption because pumps consume less power than fans.
- *Better reliability and availability:* A detailed study [3] shows that it is possible to build extremely reliable storage servers from commodity components using a high-performance communications network and software.

Implementation choices for brick architectures

Brick-based architectures allow a wide variety of implementations; some examples follow:

- Shelves of standard personal computers, each containing a switch card and software to enable the functions described in the eight criteria above.
- Racks of 1U or 2U servers, equipped as above.
- Blade servers with appropriate software (provided that switches are integrated with the blades).
- Bricks stacked to form 1D *towers*, 2D *walls*, or 3D *cubes*.

The last option deserves explanation. In a distributed system, three key design parameters are closely related: system density,[5] cooling, and intrasystem communications performance. It is easier to achieve high communications performance if the intrasystem distances are short, which is the case for dense systems. However, high-density systems imply high power densities and therefore cooling challenges. One approach is to use low-power components within dense systems, but such components invariably provide lower performance than same-generation high-power components. The Intelligent Bricks project takes a different approach by introducing a new cooling system that can handle extreme power densities [4].

Cooling is now one of the most difficult problems in modern data centers, because the heat flux of modern CMOS processor chips is as high [5] as those of the most advanced bipolar chips ever used in water-cooled mainframes. For representative high-performance systems, the heat load per square foot of system area has increased from 800 W/ft^2 for an IBM eServer* z900 (2000) to 3,000 W/ft^2 for an Egenera BladeFrame** system (2004). A study of numerous systems shows that the heat load for high-end servers has increased exponentially since 2000 [6]. The growth rate is forecast to slow down only because data centers cannot handle higher heat loads. This has profound implications for the future of the semiconductor industry, which used to measure progress primarily by improved performance. It has also greatly reduced the effective density of data centers, because racks must be spaced far apart to allow sufficient airflow. At a rack heat load of 500 W/ft^2, air-cooled racks should cover no more than 12% of the floor space, down from 30% in typical data centers today.

This difficult situation has prompted us to create a new thermal architecture, inspired by nuclear reactor designs and described in detail in the section on thermal architecture later in this paper. This allows space-filling stacking of bricks to form a 3D cube of very large dimensions (thousands of bricks). It should be emphasized that we see this effective thermal architecture as complementary to efforts to improve the energy efficiency of semiconductor devices, not as a replacement. The thermal problem facing the computer industry is so severe that the industry needs all the help it can get.

Since the bricks touch one another, wires or fibers can be eliminated for interbrick communication. This simplifies installation and maintenance and enables very high-performance communication at low cost. Details are discussed in the section on communications and couplers. Bricks must cooperate closely in order to create a reliable system. This is true in normal operation and after a brick failure. In the latter case, large amounts of state may have to be transferred. For example, if a brick contains just

[5]For a storage server, system density may be measured in units of *storage capacity/floor space*.

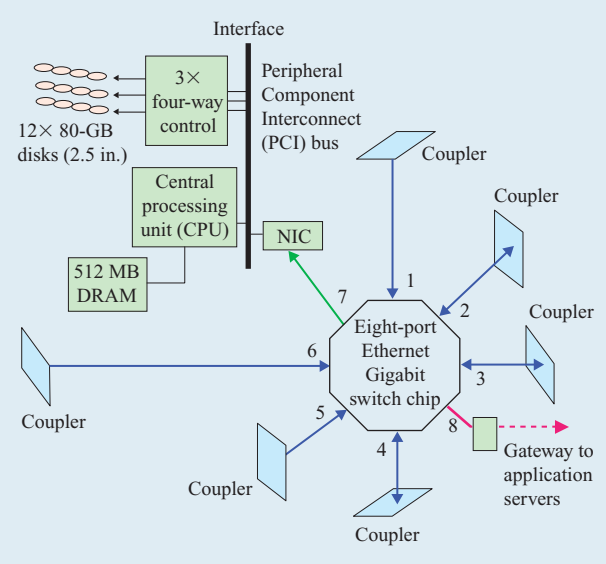


Figure 2

Electronics of one prototype brick.

1 TB of data and fails, at least the same amount of data must be transmitted within the system to create a new redundant representation of the same data. It takes 2.2 hours to transfer 1 TB over a 1-Gb/s link. During this time, the system is vulnerable to additional brick failures, with the degree of vulnerability dependent on the details of the data representation. These numbers give an impression of the enormous amount of bandwidth required to build highly reliable, distributed systems. More detail is given in the section below discussing the prevention of data loss with dRAID (distributed redundant array of independent disks). In general, a high-performance communication system makes the task of creating distributed applications much easier and less likely to fail because of problems such as congestion and timeouts. In addition, scalability over a wide range also requires high communication performance.

IceCube—prototype hardware implementation

Storage server prototype

The IceCube prototype shown in Figure 1 is a 27-brick, 26-terabyte system. It was developed by the IBM Almaden Research Center in collaboration with IBM Engineering & Technology Services (Mainz, Germany), the IBM Thomas J. Watson Research Center, and many external vendors. IceCube successfully implements the ideas presented above and should be seen, not as a protoproduct, but as a proof of concept for brick-based architecture implemented as a 3D cube.

Bricks

Following the brick architectures definition, each brick contains a switch, a central processing unit (CPU), and storage for the software and user data. The specific choices made concerning the CPU, operating system (OS), and other components are incidental, as long as the desired brick interfaces and performance requirements are met.

Figure 2 shows the electronics of each brick. At the core of the brick is an eight-port Ethernet switch chip which supports 1 Gb/s per port. Six of its ports are connected to communications devices on the surfaces of the brick. These are newly invented system-level capacitive couplers, described below. The seventh port is connected to the CPU by an Ethernet-PCI (Peripheral Component Interface) network interface chip (NIC).

It is known that Transmission Control Protocol/Internet Protocol (TCP/IP) processing in software puts substantial loads on the CPU [7]; this is in addition to the storage software load itself. Therefore, a high-performance x86 processor was chosen, supported by a chipset that was widely used in 2002 and is based on an integrated Northbridge/Southbridge controller chip (Silicon Integrated Systems SiS735). The CPU consists of 32-bit AMD Athlon** XP x86 processors with a worst-case die heat flux of about 50 W/cm^2.

In each brick are 12 laptop 2.5-in. drives, each with an 80-GB capacity, offering a total storage capacity of 960 GB per brick for user data. The software is stored on a 256-MB flash memory. The BIOS (basic input/output system) uses a modified version of Phoenix Technologies code. The 12 disks are controlled by three Advanced Technology Attachment (ATA) RAID 0/1 controllers, operating all drives as ATA masters.

The brick was designed primarily by IBM Engineering & Technology Services in Mainz, Germany. The overall function was subdivided into a passive PCI backplane board, a CPU mother/daughter board which plugs into the PCI backplane, a communications board, a 12-way disk controller board, and two power supply boards. Since two of the boards contained several PCI devices each (a situation not found in a standard personal computer), the PCI bus on the backplane is supported with an auxiliary bus which disambiguates the PCI addressing.

Figure 1(b) exposes the inside of the brick. The 12 disks, only one of which is visible, are clamped side by side between two thick aluminum plates, using thermal conducting material as a buffer. A Fourier analysis of the vibration spectra of operating disks determined that vibration coupling between the disks is barely detectable. Component cooling is described below in the thermal architecture section.

The rectangular blue structures visible on the brick surfaces are capacitive couplers, which are used to connect the brick to all of its neighboring bricks. At the top and the bottom of each brick is a standard floating power connector, which carries 48 VDC and provides wiring for a control circuit discussed later.

The dimensions of a brick are $20 \times 20 \times 25$ cm at 8.9 kg (roughly $8 \times 8 \times 10$ in. at nearly 20 pounds).

Communication and couplers

Capacitive coupling is an excellent match for brick architectures and has generated considerable interest; therefore, it is discussed in some detail.

Capacitive couplers are insulated flat metal plates mounted on all six surfaces of a brick. A pair of plates consisting of two neighboring bricks forms a small capacitor that can transmit high-speed electrical signals between bricks [8]. Using this technology with the brick architecture has brought the cost of a 10-Gb/s port down to considerably less than $100, including the per-port cost of the switch chip itself, the couplers, and the internal brick wiring. This cost is substantially less than that of conventional solutions using centralized switches and long wires or fibers.

Another reason for inventing this interbrick connection is the mechanical insertion problem. For example, if there is a hole in the top layer of bricks, one should be able to simply drop a brick into it. With conventional connectors, this would be awkward, because the connections have to be made on multiple sides simultaneously. Couplers provide an elegant solution to this problem because they slide past each other when bricks are inserted. Capacitive coupling is only one of a variety of related contactless schemes that could be used for this purpose; others include inductive coupling and electromagnetic (i.e., wireless or optical) transmissions.

The alignment of capacitive couplers is not very critical and can be aided by simple mechanical structures [9]. It is expected that one can scale this technology to support thousands of bricks without requiring expensive high-precision mechanical components.

A pair of coaxial cables transmits differential dc-balanced signals from conventional transceivers to the back of the metal plates. The transmitting and receiving paths are identical. No special amplifiers or compensation networks are required as long as the capacitances are large enough and there are no parasitic reactances. The required minimum capacitances are 70 pF for 1-Gb/s links and 25 pF for 3.125-Gb/s links.[6] Such values can readily be achieved by coating flat metal plates of less than one cm^2 area with thin dielectric layers and having the plates touch gently.

[6]It should be emphasized that high serial capacitance is desirable in a capacitive coupler. This is the opposite of the common situation in high-frequency electronics.

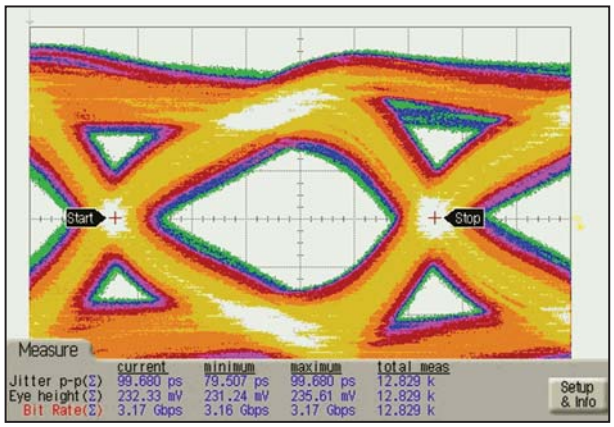


Figure 3

Eye diagram of a capacitive coupler used for 10-Gb/s transmissions.

In Figure 1(b), several capacitive coupler plates are visible. They are designed for 10-Gb/s transmissions over four parallel channels. This corresponds to the industry standards used for 10-Gb/s Ethernet, Fibre Channel, and InfiniBand**. There are a total of $16 = 4 \cdot 2 \cdot 2$ plates per coupler. This is required to support four-channel bidirectional transmission, which doubles the number of wires, and each is differential, which doubles it again. Thus, the total number of wires is $4 \times 2 \times 2 = 16$, and the total number of pads is 16. The carrier for the metal plates is a flat ceramic substrate.

The standard XAUI (10 Gb/s Attachment Unit Interface) protocol [10] for Ethernet, InfiniBand, and Fibre Channel physical interfaces employs an 8B/10B encoding scheme [11] to avoid the transmission of low-frequency signals over optical channels. Fortuitously, this also solves the low-frequency cutoff problem of capacitive coupling. The 8B/10B encoding scheme guarantees that the run length of zeros or ones is limited to five each. The 3-dB low-frequency cutoff was chosen at 1/50 of the data bit rate.

Figure 3 shows the measured eye pattern for one of the channels used with 10-Gb/s links at 3.17 Gb/s, which is slightly faster than that required for 10-Gb/s Ethernet links. The eye is wide open, as one would expect for a clear communication channel. Preliminary 3D electromagnetic field model studies indicate that the technology may be extendable to 40-Gb/s links.

IceCube base—power, cooling, and control

The collection of bricks sits on a base that provides power, cooling, and certain control functions to the bricks.

Thermal architecture

This section describes the details of the IceCube thermal architecture [4] (**Figure 4**). A vertical array of cold rails

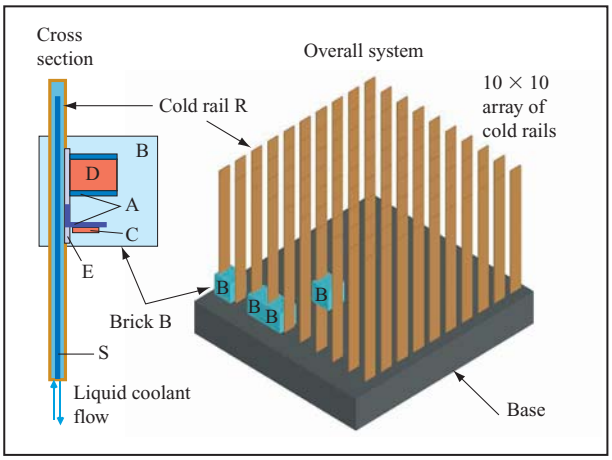


Figure 4

Thermal architecture for a 3D brick-based system.

(R) serves as the thermal backbone of the system. Bricks slide down along the rails and are clamped to the rails to make good thermal contact. Liquid coolant removes heat from the cold rail. Coolant circulates through the cold rails, is supplied to the bottom ends of the rails, and returns through tubes (S) internal to the cold rails. Bricks can be added while the system is running; this was even done during the assembly of the prototype. For lower-powered systems, the liquid coolant within the cold rails could be replaced with a one-way, upward flow of chilled air.

Note that the liquid coolant does not enter the bricks; it remains inside the cold rails. Heat is removed by dry thermal contact between bricks and cold rails, as shown on the left side of the figure. Hot elements, such as chips (C) or disks (D) within a brick (B), are in thermal contact with a thermal backplane (E). Heat-conducting elements (A), such as heat pipes or aluminum structures, carry heat from chips (C) or disks (D) to a flat aluminum plate (E), which is part of a brick and is in good thermal contact with the cold rail (R). Only the hottest chips require cooling with heat-conducting elements; the other components (up to 4 W per component) convectively dissipate heat into the air inside a brick, which itself is cooled by the cold metal structure of the brick itself. In the prototype system, the air temperature within bricks stabilized at 40°C without any use of fans, and the measured junction temperatures of the Athlon CPU chips under load are only 25–45°C.

A liquid-cooled intelligent brick system is quiet. This is an important consideration, since it is difficult for large, fan-cooled systems to adhere to the noise limits (<75 dBA) set by government regulations.

Nearly all of the heat is removed through the cold rails and very little through the surface of the bricks. This means that the cube can scale to very large dimensions, at least in the horizontal dimension. Vertical scaling is limited because the cold rail must remove all of the heat from its column of bricks. Floor loading sets another limit to vertical scaling.

It may be possible to handle power densities of several hundred kW/m^3 with practical temperature differences (~20°C) between the top and bottom of the cold rails and carefully designed brick internals. Nuclear reactor cores cool thermal power densities of approximately one hundred MW/m^3 with similar temperature differentials [12].

Power system

A 208-VAC-to-48-VDC redundant and modular power supply located in the base provides the bricks with power. The power supply can be remotely controlled via Ethernet and is interlocked with the cooling system. Conventional hot-swap floating power connectors on the top and bottom surfaces of each brick carry 48-VDC power vertically along each of the nine columns in a tapped-bus arrangement. The lower bricks in a column must carry the current for the bricks above them. The lower voltages required for the electronics are created locally within the bricks using dc/dc converters.

Power distribution at 48 VDC is suitable for (large) storage server applications, in which the total power consumption per column is of the order of 1 kW (assuming 200 W per brick, stacked four to five high). Future compute bricks, which may dissipate 1–2 kW each, will have to distribute higher-voltage power along the columns, complicating conformance with safety regulations.

Single points of failure

The base of the prototype contains several single points of failure because it was not an essential project objective to eliminate them, although this could have been done with additional engineering effort. The failure points are the integrity of the coolant distribution system, the control modules, and parts of the 208-VAC power distribution system. The 48-VDC system is redundant. A future system would contain redundant control modules, and the cooling system could be compartmentalized in various ways so that a leak would not render the system inoperative. Note that insertion or removal of a brick does not require opening the cooling system loop. There have been no problems with the IceCube cooling system in more than one year of operation.

Since all bricks in a given column share power and cooling, there is a higher probability of correlated failures for vertically stacked bricks. Software recognizes this

and avoids storing redundant data on bricks in the same column.

External connections

In this prototype system, a total of nine Gigabit Ethernet connections to external application servers was deemed sufficient. The capacitive couplers on the vertical faces of nine bricks on the surface of the cube were replaced with input/output pods, as shown in Figure 1(a). These connect application servers to the otherwise unused eighth port of the Gigabit switches in these bricks. Seven blue light-emitting diodes (LEDs) are controlled by software and help the user to visualize traffic through the surface bricks.

Industrial control modules are mounted in the base. They are used to sense critical system voltages and temperatures, detect moisture, and measure the coolant flow rates in the nine cold rail loops; under control of an external management processor, they can detect the presence of individual bricks and turn them on or off. The control modules utilize a point-to-point signaling path to perform the presence-detect and power-control functions. This is implemented very robustly by directly controlling the dc/dc converters without any assistance or interference from the brick processors or software. Thus, an errant brick (e.g., one that floods the network with packets and does not respond to any inputs) can always be shut down or reset. The point-to-point path uses additional pins on the floating connectors of the dc power distribution system.

System reliability—deferred maintenance and fail-in-place

Overview

The brick architecture as defined herein requires that, if any brick fails, the system will continue to function without noticeable impact. This is particularly important for 3D systems, in which replacement of bricks is impractical.

Very large systems must use a failure-tolerant architecture. Petascale systems contain millions of electronic components, and the chance that they are all working at the same time is negligible [13]. Once a system is sufficiently failure-resilient, maintenance may be deferred. This feature is highly valued by customers because it eliminates the cost of ongoing maintenance and avoids the potential for human error when performing the maintenance. According to IBM internal studies, the latter has become a very significant source of system outages. The ultimate goal is to build a system that requires no maintenance during its entire lifetime. Whether or not this goal can be achieved depends on both the reliability of system components and the correctness of the software managing it.

A companion paper [3] in this issue presents a detailed analysis of fail-in-place and deferred maintenance. This section presents only a summary of key results.

Preventing data loss via dRAID

The implementation of deferred maintenance for storage servers requires that data be replicated over multiple bricks so that the failure of one (or more) disks or bricks does not lead to loss of user data. The algorithm for placement of redundant copies of data over multiple bricks is called *distributed RAID* (dRAID) [14].

The following example illustrates the basic idea of dRAID. It assumes that a file is mirrored on brick A and brick B. If brick A fails, the file can still be retrieved from B. However, if B fails, the data will be lost. Therefore, after the failure of A, system software is required to copy the file (while maintaining coherency in the face of concurrent updates to the data) from good brick B to a third good brick C. There is a time window of vulnerability after the failure of A and before the copy operation from B to C is completed. If the file is very valuable, it can be stored on more than two bricks, thus further reducing the probability of data loss.

A simple N-way mirroring (i.e., more than two copies of data are created) is not a good tradeoff between cost and reliability, but performs very well when writing. Various RAID schemes are in wide use today, in particular RAID 5, in which the exclusive-OR of all data on N disks is written onto an additional disk [15]. This allows reconstruction of data if one of the $N + 1$ disks fails. The brick architecture, with its distributed computing power and high-bandwidth interconnect, enables many versions of dRAID. Its four goals are to accommodate the addition or loss of bricks, to allow the storage administrator or user to determine the tradeoff between cost and probability of data loss, to maintain acceptable performance, and to use system-wide spare capacity efficiently. It is possible to achieve extraordinarily low probabilities for data-loss events with capacity penalties comparable to those for two-way mirroring, even while assuming the high failure rates of commodity disk drives [16]. The target reliability is only two data-loss events per exabyte-year due to multiple failures.

The high-bandwidth interconnect of the IceCube mesh is the key enabler for these dRAID algorithms. It provides the flexibility to use spare capacity from anywhere in the cube. This prevents a common problem with conventional RAID controllers—i.e., the statically preallocated spare disk capacity for each disk array is small and fixed, making it very difficult to achieve above-target reliability unless failed disks are quickly replaced.

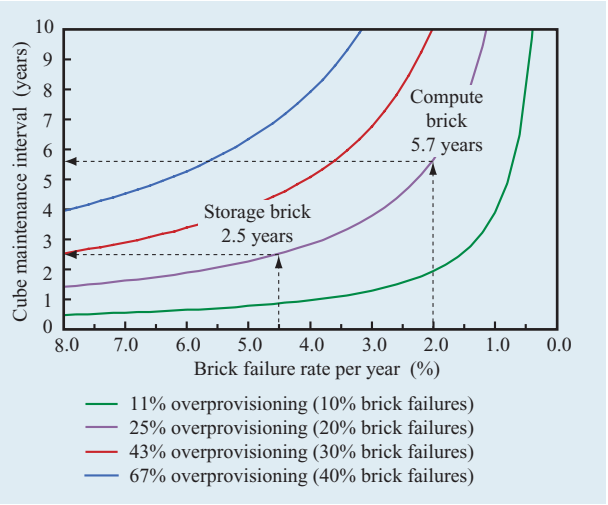


Figure 5

Required maintenance intervals as a function of brick reliability.

In such a high-pressure repair situation, there is also a significant probability of replacing the wrong disk, which itself is a leading cause of data loss.

Brick failures in a 3D mesh

This section presents a summary of the effect of brick failures. A detailed analysis is found in [3]. The analysis ignores failures in the IceCube base and errors in the storage software.

The system can tolerate the failure of numerous bricks; only after 40% of all bricks have failed is a nonlinear degradation of performance (bandwidths, usable capacity, and I/O) observed.

These findings are surprisingly insensitive to the details of the system studied, such as the overall size of the system and the level of the storage redundancy chosen. For optimum performance, a 3D system should be operated with about 70% of its bricks operational. Another conclusion from the detailed analysis is that external application server systems should be connected to multiple surface bricks, either directly or through an external switch.

The results, however, are quite sensitive to the dimensionality of the cube. A 2D mesh requires that about 85% of the bricks be operational. This is not surprising, because there are fewer redundant communication paths in a 2D system than in a 3D system.

Reliability projections and lifetime estimates

This section discusses the number of spare bricks that must be provided in order to keep a system operational for a given period of time without requiring maintenance (adding more bricks). This depends on the hardware failure rate of brick electronics and disks. The results, assuming that the system requires a *five-9* (0.99999) probability of being available, are shown in **Figure 5**. The parameters of the curves are percentages of overprovisioning. If one assumes a rather typical hardware failure rate of 4.5% per storage brick per year, equally split between electronics and disks, a system requires an overprovisioning of 25% to support a maintenance-free lifetime of 2.5 years. While 25% overprovisioning is high, more realistic scenarios call for a much lower number. If a customer buys spare bricks on demand (that is, only when the actual free capacity in the system falls below a certain level), the total number of spare bricks found in the system after a few years is much lower; this is because the newer bricks have a higher storage capacity.

Software

The software currently running on IceCube provides a distributed, scale-out file system. The long-term goals are to provide a reliable high-performance file service, decrease administrative costs, and increase system efficiency.

For the software discussion, it is assumed that the system is used as a storage server. The present state and near-term plans for the software are discussed. In a role as storage server, no end-user application software is expected to run on the brick processors; only a restricted set of open-source and IBM-owned programs are used. End-user applications run on compute-oriented servers, such as Blue Gene*/L [17], which are collectively called *application servers*.

Major components

The software includes four distinct elements: a distributed file system for storing data in the bricks, a monitoring and control system for safety and power control of the hardware modules, a self-management system for analyzing system state and performing recovery actions for hardware and software problems, and a user interface for configuration and reporting.

The system software residing on IceCube and its attached application servers is shown in **Figure 6**. The system software includes software running on a management processor connected to the IceCube base in addition to the software running in each of the IceCube bricks.[7] The OS, the mesh routing protocol software, and a thin layer of IceCube-specific low-level modules that provide hardware monitoring and control together form the operating environment.

[7]The prototype has a single management processor, which is a single point of failure. A larger system would employ *N* management processors with a fail-over scheme.

Two major pieces of user-level code reside on top of the operating environment. These are the IBM General Parallel File System (GPFS) [18] and Kybos (the companion research project building the software for the IceCube prototype). GPFS is a clustered file system product developed originally for the IBM SP* family of parallel supercomputers and has been deployed on SP systems with up to 2,000 nodes. Kybos provides the self-management software and a management interface to the administrator's Web browser.

Operating environment

The operating environment provides the low-level software platform on which distributed self-managing storage services run. All executable software is stored on a flash memory inside each brick. Each flash contains two different versions of the software. Thus, if a new code load renders a brick inoperative, one can revert to the previous version. For performance reasons, the executable software is copied onto disks during boot.

Linux** is used for both the bricks and the management processor. Certain drivers for the specific brick hardware (that is, the NIC and the eight-port switch) have been built into the Linux kernel (v2.4) used in the bricks.

As shown in **Figure 7**, the management processor interfaces directly with the industrial control modules in the base, which connect to thermal, water flow, and brick presence sensors and also drive power control signals to each brick. Because of this direct connection, the management processor has a powerful tool for restoring system stability after failures are detected.

Each brick individually monitors processor and drive temperatures and reports the data back to the management processor using a Linux cluster management tool called Ganglia [19]. The management processor runs software that monitors and records the brick states in a database. This happens approximately once per second and also serves as a heartbeat detector for the bricks. This database is read by the Kybos software, which takes appropriate action (as described below in the Kybos section) and displays the current state of the bricks, as shown in **Figure 8**. The main part of the screen shows performance statistics gathered from the bricks. The window in the left corner shows overall status and parameters for the cube, such as voltages, on/off status,

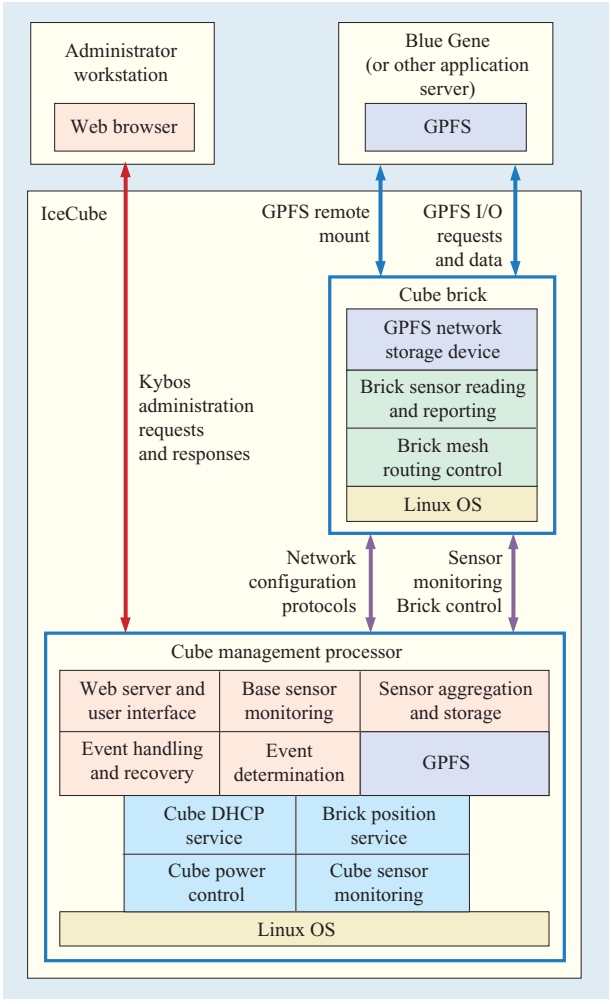


Figure 6

Software stack implemented on IceCube.

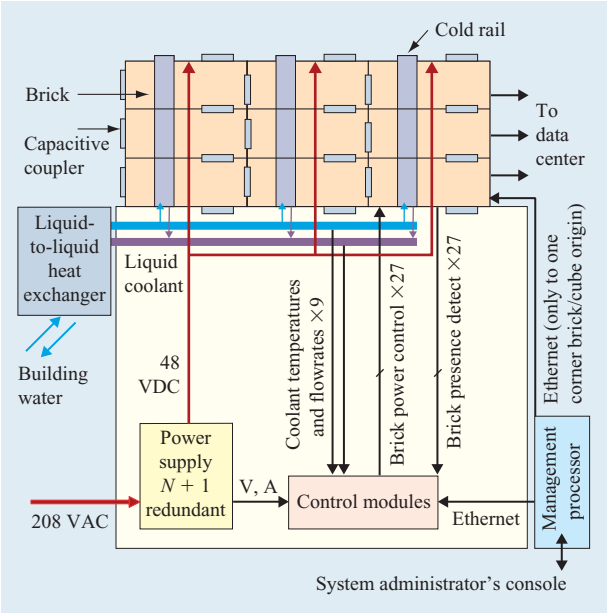


Figure 7

Block diagram of the IceCube base.

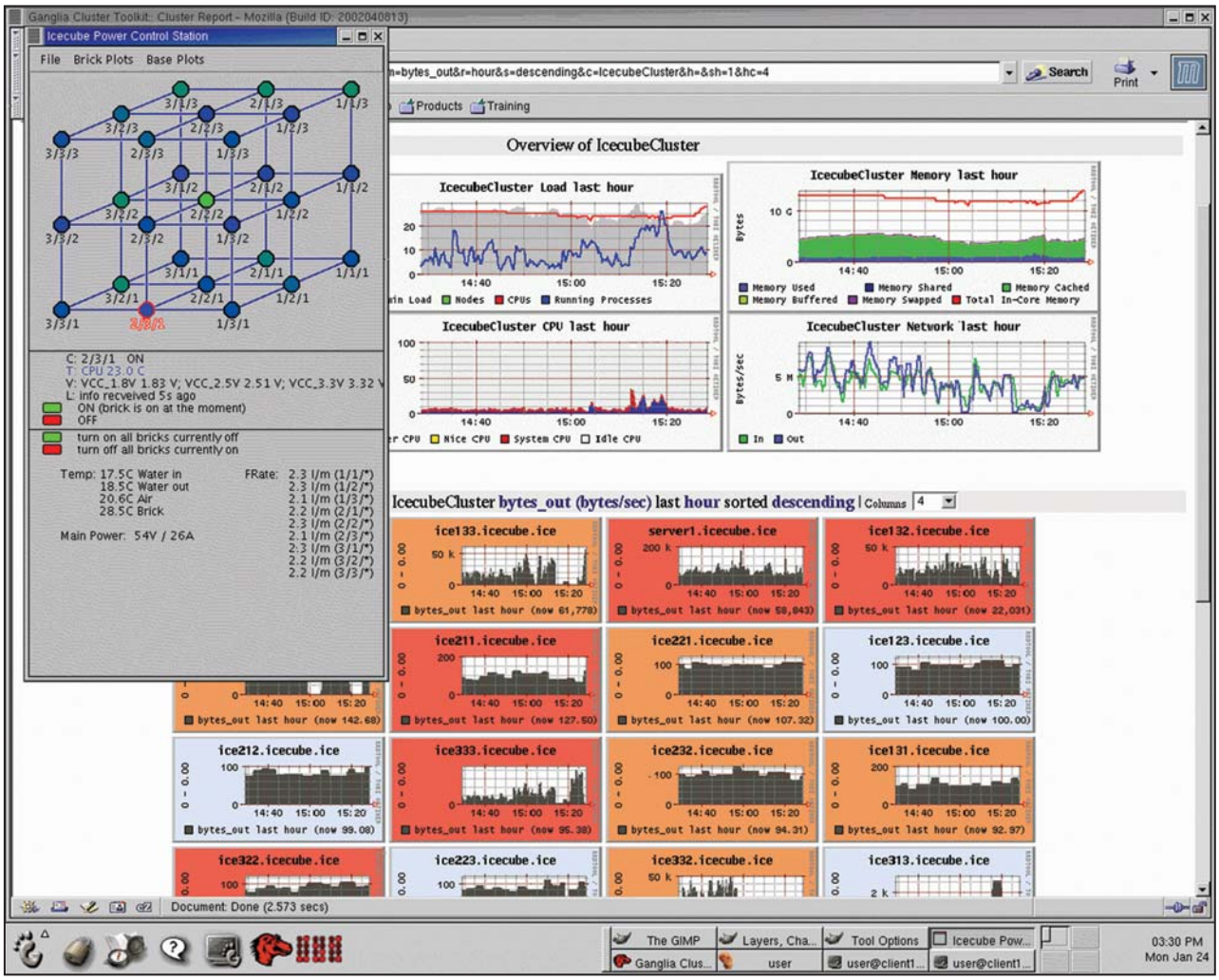


Figure 8

Control panel for the IceCube operating environment.

and coolant flow rates. Sensor data from within one brick, which is selected and highlighted with an orange circle and labeled "2/3/1," is displayed.

Network communications

The switch chip within each brick must be supplied with a routing table. The routing table depends on the topology of the system, including the cube itself and external switches and application servers. This has been accomplished with the IceCube mesh routing control software. The commonly used spanning tree algorithm is unsuitable for a highly connected 3D topology with loops. Instead, an algorithm determines possible communication paths between pairs of nodes and selects one with the minimum hop count. The selected paths remain in use until the topology of the mesh changes by the addition or failure of a brick. Possible improvements include the use of several shortest-distance paths in parallel. Note that there is a danger of forming loops if there are connections to an external switch. Special care is taken to ensure that messages that enter the cube and are not targeted for a specific brick (such as broadcast messages) are intercepted before they leave the cube through another brick.

Ethernet connects the management processor directly to a brick in one corner, which serves as the origin of the internal Cartesian coordinate system of the cube. The position server process in the management processor provides that service, and each brick can determine its position with reference to the origin.

A Dynamic Host Configuration Protocol (DHCP) server in the management processor issues IP addresses to

all bricks, enabling TCP/IP and UDP (User Datagram Protocol) communication between bricks and between bricks and application servers.

GPFS

GPFS provides the basic I/O service layer for the system. As seen in Figure 6, the IceCube GPFS cluster has services running in the management processor and in the bricks. The management processor GPFS node serves as the sole quorum node for the IceCube GPFS cluster. GPFS services running in the bricks are therefore not required for a quorum (i.e., most bricks could fail and the GPFS cluster would continue to operate). The major GPFS function running on all of the bricks is the network storage device (NSD) service. This GPFS service exports the disks in the bricks as logically shared disks, instructing GPFS that all I/O requests for those disks should be directed to the NSD service exporting the disk.

To access IceCube storage, Blue Gene/L [17] or other application servers must install a GPFS cluster (separate from the IceCube GPFS cluster), in which a GPFS node runs on each Blue Gene/L node that requires access to IceCube storage. The Blue Gene/L GPFS cluster remotely mounts an IceCube GPFS file system, allowing it to make file I/O requests to the IceCube file systems. When such requests are made, the file requests are mapped by the Blue Gene/L nodes to NSD nodes in the IceCube GPFS cluster, and the file requests are sent directly to the appropriate bricks from the Blue Gene/L nodes. The management processor is not involved in this process.

GPFS includes sophisticated controls for disk management, including performing data and metadata striping, mirroring, rebalancing, and migration. The Kybos management software leverages those controls as described in the next section.

Kybos

The purpose of the Kybos software is to reduce the administrative effort required to maintain a storage system. Its management effort scales with the number of applications using the storage system rather than with the number of disks in the system, as in many storage systems today. It is based on a policy-oriented management interface. An administrator classifies sets of data for each application according to a small number of clear and simple service goals, along with rules for the identification of newly created data (by directory, by user ID, by file name, by file type, etc.).

Kybos implements a self-management control system that performs three functions. First, it monitors the state of the system as reported by the hardware sensors described earlier and the state of the file system as reported through the GPFS administrative interface. Second, it detects important events by analyzing the current state and trends of the system against the data service goals set by administrators, along with other system invariants (such as routing statistics and acceptable voltage and temperature ranges). Third, it schedules activities (such as data migration, restoring of desired data redundancy, and replication or backup of sets of data) to realign the system state with the data service goals.

In the Kybos model, the system administrator expresses the goals for a given set of data by creating a Kybos *resource pool*, described in terms of capacity (lower reserve and upper limit on the number of gigabytes of storage), performance (lower reserve and upper limit on the throughput and/or response time), and reliability (e.g., for the active copy, the number of data losses per exabyte-year of stored data that are acceptable, and for GPFS secondary copies, a recovery-time objective). Kybos remembers the goals for resource pools, and when a new file is created, identifies the resource pool to which the new file is to be assigned according to the attributes of the file. It then determines how the file is to be placed on physical disks.

Kybos self-management relieves administrators of other device-oriented management tasks as well. Today, for example, it requires many distinct steps to install additional physical storage and make it visible in a typical storage area network (SAN)-based storage server. The steps include gathering requirements, installing the disks, controllers and cables, configuring the RAID arrays and virtual disks, host mapping, zoning, initiating the rediscovery of SAN devices in the application servers, and extending the file system. As an example, in a case study, 25 person-days were required to perform these tasks. Using the Kybos management model, the time was estimated to be reduced to seven person-days, of which most were required for high-level planning. This is because it is expected that the administrator simply needs to physically place the bricks in the cube and define the rather obvious resource pool parameters. This leaves Kybos to discover, power, and use the bricks.

System status

The prototype cube is connected to a two-rack Blue Gene/L system at the IBM Almaden Research Center via Gigabit Ethernet links and other application servers. The mesh routing control protocol has been implemented and demonstrated to work in scenarios with multiple failures. File storage and access from Blue Gene/L using GPFS and hardware monitoring and visualization are completed and have been demonstrated.

The first integrated version of Kybos self-management software is currently being implemented. It provides resource pools with the ability to set coarse-grain capacity

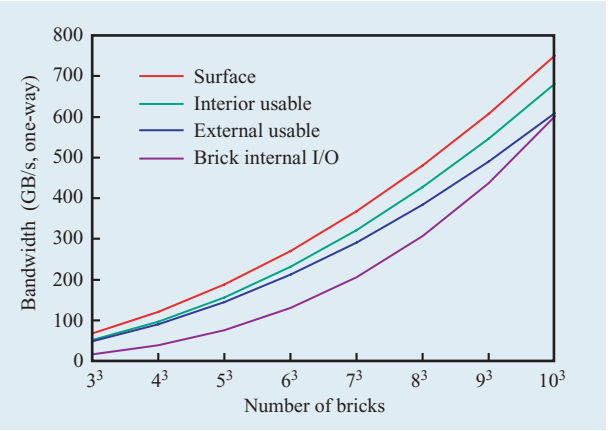


Figure 9

Calculated bandwidths as a function of IceCube dimensions.

goals and limited reliability goals (mirroring/no mirroring of primary data copy). The first version will also implement all hardware monitoring and safety controls for the prototype and sufficient GPFS monitoring and control to detect failed components and invoke recovery actions in GPFS. A more extensive set of self-management algorithms is also being developed on the Kybos simulation platform.

Other file services

Thus far, the sole storage access method used has been via GPFS remote mounting of the IceCube GPFS file systems, although there are other access methods that can be deployed. For example, selected bricks could also run Network File Service (NFS) servers, or all bricks could run distributed NFS servers. Other IBM projects at Almaden Research are building such capabilities. Common Internet File Service (CIFS) is another option that could be implemented on the system. Note that the cube can, in principle, run any software developed for a Linux cluster, because that is what it is.

Performance

The following section summarizes the key topological properties of a 3D mesh architecture. We ignore the fact that some of the nodes may be broken, since the performance implications of failing bricks are studied in another paper [3] in this issue.

For a symmetric cube with $N = h^3$ nodes connected by links of bandwidth **z**, the following relations hold [The numbers in parentheses are examples for a symmetric pristine system with $h = 10$, $N = 1{,}000$ nodes using links with $z = 1.25$-GB/s (one-way) bandwidth.]:

- Total cube bandwidth: $BW_{\text{peak}} = 6 \cdot N \cdot z$ (7.5 TB/s).
- Cross-sectional cube bandwidth: $BW_{\text{cross}} = k^2 \cdot z$ (125 GB/s).
- Average number of hops (for random traffic inside the cube): $d_{\text{avg}} = h - (1/h)$ (9.9 hops).
- Usable interior cube bandwidth (random node to random node traffic): $BW_{\text{usable}} = BW_{\text{peak}}/d_{\text{avg}}$ (760 GB/s).

Figure 9 shows various bandwidths as a function of cube size. Surface bandwidth is the total bandwidth (assuming 10-Gb/s links) through the surface of the cube. Interior bandwidth is the total usable interior bandwidth from a random brick to another random brick. Note that this is not simply the sum of the bandwidths of all bricks; it is reduced by the fact that a given link inside the cube has to carry the superposition of all traffic between all node pairs whose communication paths include that link (see the usable cube bandwidth equation above). The external bandwidth is the total bandwidth through the surface of the cube to random destination bricks inside the cube. The brick internal I/O bandwidth is the sum of the bandwidths between the I/O devices inside a brick, for all bricks, taking into account bus traffic effects and disk limitations.

By any measure, the numbers for big cubes are very large and are sufficient to provide I/O for the fastest supercomputers. For example, the I/O bandwidth required by the 360-teraflops IBM Blue Gene/L machine under construction for Lawrence Livermore Laboratory could be provided by only a $6 \times 6 \times 6$ cube. Although the ratio between surface area and volume decreases linearly with increasing cube size, the ratio between surface bandwidth and internal usable bandwidth remains nearly constant. This can be seen from inspecting the equations above, and is due to the superposition of traffic discussed earlier.

Note that the prototype IceCube system is built with integrated circuits that were first introduced several years ago. Thus, the prototype system is about an order of magnitude slower—for all interesting parameters—than what one could build today with the same number of bricks. Nevertheless, even the IceCube prototype is quite a capable system. For example, it could store 5,000 movies in DVD format and simultaneously stream 900 MPEG-2 streams to users. These numbers are based on measured sustained data transfer rates to application servers.

Future

Very large systems

The following section discusses the scalability of the IceCube architecture to very large systems. As discussed

earlier, thermal and power constraints are not an important issue for this architecture. A critical concern is floor loading. Typical limits today are 500–1,200 kg/m^2 (100–250 lb/ft^2). This effect limits the number of bricks that can be vertically stacked and forces large systems to spread out horizontally. Eventually, this will change the average distance messages must travel between nodes from scaling with the third root of the number of bricks (for cubes) to the second root (for a flat mesh), which is a less favorable scaling. Whether or not this becomes an issue depends on the application. A storage server will be more latency-tolerant (in terms of hop count) than a supercomputer doing more closely coupled computations. Systems with 1,000 to 2,000 bricks (e.g., five or six bricks high and 15 to 20 bricks wide and deep) appear to be a practical upper limit. This corresponds to a dynamic scaling range of two orders of magnitude if one assumes that the smallest practical system contains ten bricks.

However, it would be impractical to provide a single base of such a size. Rather, one could partition the system into smaller subcubes and assemble the system *in situ* from these subcubes. It is likely that one could build self-aligning capacitive couplers that could tolerate centimeter-scale misalignments between subcubes. Such couplers would remove the need to connect subcubes via cables or fibers; rather, the same 3D mesh structure could be extended across the entire system. This would greatly simplify the assembly of very large systems.

Commodity serial ATA 3.5-in. disk drives should approach one-terabyte capacity in 2007. A 1,000-brick system, each brick containing eight such drives, provides eight petabytes (PB) of raw storage capacity. The use of a strong dRAID algorithm reduces the usable storage capacity to 4–6 PB [16], with a data-loss probability resulting from simultaneous failures measured in a only a few events per exabyte-year. This is an interesting system for many emerging applications that require storing large amounts of data, such as images, digital media, and regulatory compliance data.

Small systems

The brick architecture as described is designed for medium-sized to very large systems, where the failure of a few bricks makes little difference. Nevertheless, the ease of management and scalability promised by the architecture has prompted inquiries about its applicability to systems for the small and medium-sized business market, which require only a small number of bricks. For systems with fewer than approximately eight bricks—in which the failure of a single brick becomes noticeable—the brick design should be modified to include brick-internal RAID and possibly a provision to hot-swap bricks.

Petaflops compute servers

This project was conceived as an architecture for petaflops supercomputing, not as a storage server. Seymour Cray [20] once said: *It's the heat and the thickness of the (wiring) mat which matters*. That is still true today, and the Intelligent Bricks architecture directly addresses these two issues. Very powerful processors can be used because they can be cooled. The architecture also allows scaling to large numbers of nodes and provides high, low-cost bandwidth between nodes. Communication latency, because of the multihop architecture, is larger than for centralized switch architectures, but 15 years of experience with message-passing supercomputers has shown that latency is dominated by software latency at the endpoint nodes—not by hardware communication latency [21].

A more detailed analysis of performance ratios based on traditional ratios between supercomputer system parameters [such as floating-point operations per second (flops), total memory size, bus bandwidths within bricks, I/O bandwidth, and interbrick communication bandwidths] shows that it is possible to build well-balanced supercomputers in the petaflops range with this architecture.

There is one important issue with the fail-in-place assumption for compute applications: Unlike storage servers, supercomputers run a wide variety of software that is provided by users, not the system vendor. If an application assumes that allocated resources will never go away, it cannot deal with failing nodes without aborting and remapping the problem. Fortunately, a growing part of the information technology industry is dedicated to grid computing. They must solve exactly the same problem, and any solution found will be directly applicable to brick-based supercomputers.

Related work

The approach of distributing data across independent nodes to build scalable storage systems has been explored by both academic and commercial projects. These include DataMesh [22], FAB [23], Self-* [24], Petal [25], and OceanStore [26], which are primarily research projects.

To the authors' knowledge, no company has realized the 3D brick packaging as described here. However, numerous companies are working on various forms of distributed enterprise storage. Among these companies are Panasas [27] (scalable storage for Linux clusters), Isilon (distributed file system on standard hardware), Pivot3, Pillar Data Systems (storage management), Lefthand Networks (iSCSI IP-based SAN software), Equallogic (iSCSI-based systems), Ibrix (storage software suite for the enterprise), Cluster File Systems (Lustre** open source object store software), Google (GFS), and Archivas (software for reference storage).

Conclusion

The Intelligent Bricks project is demonstrating the feasibility of brick architectures. The exact meaning of this term is defined in this paper. The salient feature is not the physical shape of the packaging units, but rather the biologically inspired model that any brick in a system, like a cell in an organism, may fail without noticeably affecting the operation of the system as long as most bricks are functioning normally. The required repair actions are done by system software, without human intervention.

This model has numerous positive consequences. It simplifies system management, eliminates common system failures caused by inappropriate human intervention, and allows highly efficient packaging with a concomitant improvement in important parameters such as scalability, communications performance, system density, and thermal management.

A working prototype of such a system has been built at the IBM Almaden Research Center and performs as expected. It implements a 27-brick, 26-TB storage server. The system software is based on Linux (the OS for each processor), GPFS (the distributed file system), and Kybos (the management software specifically developed for the project). While it is purely a research project, extensive external exposure to IBM customers has yielded strong confirmation that the objectives of the project are well aligned with market needs.

Acknowledgments

This work has been funded by the IBM Research Division over a period of several years. The authors especially thank Drs. Jai Menon, Dilip Kandlur, Robin Williams, Eric Kronstadt, and Robert Morris for their long-term support for this project.

We extend our profound thanks to Jim Speidell of the IBM Thomas J. Watson Research Center, Dave Altknecht of the IBM Almaden Research Center, Karl-Heinz Lehnert, IBM E&TS, Mainz, and Bob Steinbugler for their strong management support, and to Dr. Gaby Persch-Schuy, IBM E&TS, Mainz. We thank Mike Rogers, Vincent Arena, Andy Perez, and Ron Ridgeway for their work related to the design and manufacturing of the brick electronics and Aaron Cox, IBM Tucson, for the industrial designs. Dr. Roger Schmidt of IBM Poughkeepsie was the source of many thermal ideas. We thank Larry Mok, Mark Ritter, and Al Widmer from the IBM Thomas J. Watson Research Center and Professor Martin Graham of the University of California at Berkeley for many enlightening discussions.We also thank our intern, Ekpe Okorafor, who worked on the base software.

We express our most special thanks to Mr. Ikuo Suesada, Chairman and CEO, Mr. Shingo Shimada, Chief Executive Director and Vice Chair of NIWS Company, Ltd., Tokyo, and members of their staff for their support.

Several companies contributed their expertise to the project, including Applied Engineering (F. Zadeh), InterCon Systems (G. Buschbaum), Rock Solid Design (E. Susbilla and D. Bailey), PCB and More (J. Freese), Sonic Manufacturing (H. Woo), Thermacore (W. Cho), Vibren Technologies (L. D'sa and M. Nottage), Rick Williams Consulting, and Productware (M. Bolle). We also thank the following vendors for their superlative technical support: AMD, Broadcom, Opto22, Ericson Power, Valere Power, KOK Technologies, Proto Engineering, Streamline Circuits, AIK Laminates, M & W Systems, Noren Products, Mindspeed Technologies, Motorola, Amphenol, Agilent Technologies, GrafTech, Bergquist, and Chomerics.

References

1. N. Allen, "Don't Waste Your Storage Dollars: What You Need to Know," *Research Note*, Gartner Group Inc., Stamford, CT 06904, March 2001.
2. L. Wood, "The Hidden Costs of Unmanaged Storage," *Enterprise Storage Forum.Com*, July 7, 2003; see *http://www.enterprisestorageforum.com/technology/features/article.php/11176_2231701_2*.
3. C. Fleiner, R. B. Garner, J. L. Hafner, KK Rao, D. R. Kenchammana-Hosekote, W. W. Wilcke, and J. S. Glider, "Reliability of Modular Mesh-Connected Intelligent Storage Brick Systems," *IBM J. Res. & Dev.* **50**, No. 2/3, 199–208 (2006, this issue).
4. K. Fernandez, C. Fleiner, R. Garner, H. Huels, M. Ries, and W. Wilcke, "System and Method for Providing Cooling in a Three-Dimensional Infrastructure for Massively Scalable Computers," U.S. Patent Application 20050152114, filed July 14, 2005.
5. R. Schmidt, "Liquid Cooling is Back," *Electron. Cooling* **11**, No. 3, 34–38 (August 2005).
6. *Datacom Equipment Power Trends and Cooling Applications*, M. Geshwiler, Ed., American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., Atlanta, GA, 2005; ISBN 1-931862-65-6.
7. P. Sarkar, K. Voruganti, K. Meth, O. Biran, and J. Satran, "Internet Protocol Storage Area Networks," *IBM Syst. J.* **42**, No. 2, 218–231 (April 2003).
8. R. B. Garner, W. Wilcke, B. Rubin, and H. Kahn, "A Scalable Computer System Having Surface-Mounted Capacitive Couplers for Intercommunication," U.S. Patent Application 20040066249, filed October 2002.
9. W. Wilcke, R. Williams, R. Garner, and C. Fleiner, "Mechanism for Self-Alignment of Communication Elements in a Modular Electronic System," U.S. Patent Application 10-987,901, filed November 12, 2004.
10. J. D'Ambrosia, S. Rogers, and J. Quilici, "XAUI—An Overview," white paper; see *http://www.techonline.com/community/related_content/21173*.
11. A. X. Widmer and P. A. Franaszek, "A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code," *IBM J. Res.*

& Dev. **27**, No. 5, 440–451 (1983); see also *http://www.interfacebus.com/Definitions.html*.
12. S. Glasstone, *Principles of Nuclear Reactor Engineering*, D. Van Nostrand Company, New York, 1955.
13. T. Sterling, P. Messina, and P. H. Smith, *Enabling Technologies for Petaflops Computing*, The MIT Press, Cambridge, MA, 1995; ISBN 0-262-69176-0.
14. K. Rao, J. Hafner, and R. Golding, "Reliability for Networked Storage Nodes," *Research Report RJ-10358*, IBM Almaden Research Center, San Jose, CA 95120, September 2005.
15. D. A. Patterson, G. Gibson, and R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," *Proceedings of the ACM SIGMOD International Conference on Management of Data,* June 1988, pp. 109–116.
16. J. L. Hafner, "WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems," *Proceedings of the 4th USENIX Conference on File and Storage Technologies*, December 15, 2005.
17. *IBM Journal of Research and Development*, Vol. 49, No. 2/3, 2005; special issue on Blue Gene; see also *http://www.research.ibm.com/bluegene/*.
18. IBM Corporation,"General Parallel File System," 2005; see *http://www-03.ibm.com/servers/eserver/clusters/software/gpfs.pdf*.
19. M. L. Massie, B. N. Chun, and D. E. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience," *Parallel Computing* **30**, No. 7, 817–840 (July 2004); see also *http://ganglia.sourceforge.net/*.
20. C. G. Bell, J. C. Mudge, and J. E. McNamara, *Computer Engineering; A DEC View of Hardware Systems Design*, Butterworth–Heinemann, Newton, MA, 1978.
21. G. Bell and J. Gray, "What's Next in High-Performance Computing?," *Commun. ACM* **45**, No. 2, 91–95 (February 2002).
22. C. Chao, J. Wilkes, D. Jacobson, B. Sears, R. M. English, and A. A. Stepanov, "DataMesh Architecture 1.0," *Technical Report HPL-92–153*, Hewlett-Packard Computer Systems Laboratory, Palo Alto, CA 94304, December 1992.
23. Y. Saito, S. Frølund, A. Veitch, A. Merchant, and S. Spence, "FAB: Building Distributed Enterprise Disk Arrays from Commodity Components," *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2004, pp. 48–58.
24. G. R. Ganger, J. D. Strunk, and A. J. Klosterman, "Self-* Storage: Brick-Based Storage with Automated Administration," *Technical Report CMU-CS-03-178*, Carnegie Mellon University, Pittsburgh, PA 15213, August 2003.
25. E. K. Lee and C. A. Thekkath, "Petal: Distributed Virtual Disks," *Proceedings of the 7th International Conference for Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 1996, pp. 84–92.
26. J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage," *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2000, pp. 190–201.
27. D. Nagle, D. Serenyi, and A. Matthews, "The Panasas ActiveScale Storage Cluster—Delivering Scalable High Bandwidth Storage," *Proceedings of the ACM/IEEE Supercomputing Conference*, 2004; see *http://www.sc-conference.org/sc2004/schedule/pdfs/pap207.pdf*.

Received July 5, 2005; accepted for publication August 18, 2005; Internet publication February 22, 2006

Winfried W. Wilcke *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (winfriedwilcke@us.ibm.com)*. Dr. Wilcke is a program director at the IBM Almaden Research Center, where he started the Intelligent Bricks project and IceCube implementation in 2001. He was a Senior Manager at the IBM Thomas J. Watson Research Center, responsible for Victor/Vulcan research, later commercialized in IBM SP supercomputers. As Director of Architecture (later Chief Technical Officer) of HaL computers, he was deeply involved in the creation of the 64-bit SPARC** architecture. Dr. Wilcke received a Ph.D. degree in nuclear physics and previously worked at the University of Rochester and at the Los Alamos and Lawrence Berkeley National Laboratories.

Robert B. Garner *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (robgarn@us.ibm.com)*. Mr. Garner received his M.S.E.E. degree from Stanford University. Before joining the IBM Research Division in 2001 to work in the IceCube project, he worked at Xerox Palo Alto Research Center, Sun Microsystems, and Brocade Communications.

Claudio Fleiner *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (fleiner@us.ibm.com)*. Dr. Fleiner received a Ph.D. degree in computer science from the University of Fribourg, Switzerland. Prior to joining the IBM Almaden Research Center, he worked at the IBM Zurich Research Laboratory and Transmeta. Dr. Fleiner's research interests include large storage systems, distributed computing, and computer networks.

Richard F. Freitas *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (freitas@almaden.ibm.com)*. Dr. Freitas received a Ph.D. degree in electrical engineering and computer science from the University of California at Berkeley. Dr. Freitas is currently exploring the use of nonvolatile solid-state memory technology for storage systems.

Richard A. Golding *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (rgolding@us.ibm.com)*. Dr. Golding received a Ph.D. degree for work on weak-consistency distributed systems. He was previously an architect at Panasas and a researcher at Hewlett-Packard Laboratories. He currently leads the Collective Intelligent Bricks software effort.

Joseph S. Glider *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (gliderj@almaden.ibm.com)*. Mr. Glider is a Senior Technical Staff Member and manager in charge of the Intelligent Bricks Storage Software project at the IBM Almaden Research Center. He received a B.S.E.E. degree from Rensselaer Polytechnic Institute. Mr. Glider's research interests include distributed storage systems and fault-tolerant storage systems.

Deepak R. Kenchammana-Hosekote *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (kencham@us.ibm.com)*. Dr. Kenchammana-Hosekote received a Ph.D. degree in computer science from the University of Minnesota. He has been at the IBM Almaden Research Center since 1997 working on various RAID and distributed storage systems.

James L. Hafner *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (hafner@almaden.ibm.com)*. Dr. Hafner received a Ph.D. degree in mathematics from the University of Illinois. He later joined the IBM Research Division and has worked in diverse areas, including number theory, complexity theory, image databases, and storage-system protocols. Dr. Hafner currently works in the advanced RAID project.

K. Moidin Mohiuddin *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (moidinkm@us.ibm.com)*. Dr. Mohiuddin received a Ph.D. degree in electrical engineering from Stanford University in 1982 and has been with the IBM Research Division since that time. He is currently Senior Manager of Advanced Storage Systems at the IBM Almaden Research Center, which includes the Intelligent Bricks project.

KK Rao *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (kkrao@us.ibm.com)*. Mr. Rao is a Distinguished Engineer in storage systems. He received B.S.E.E. and M.S.E.E. degrees from the Indian Institute of Technology, Bombay, joining the IBM Research Division in 2002. Mr. Rao has been working on advanced RAID algorithms, reliability of distributed storage systems, and scale-out storage systems.

Ralph A. Becker-Szendy *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (ralphbsz@almaden.ibm.com)*. Dr. Becker-Szendy received physics degrees (Dipl.Physiker) from the RWTH Aachen University, Germany, and a Ph.D. degree from the University of Hawaii. Before joining the IBM Research Division in 2001, he worked at the Stanford Linear Accelerator Center and Hewlett-Packard Laboratories. Dr. Becker-Szendy works on distributed storage (the StorageTank file system) and storage performance.

Theodore M. Wong *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (theowong@us.ibm.com)*. Dr. Wong received a B.A. degree from Oxford University, an M.E. degree from Cornell University, and a Ph.D. degree from Carnegie Mellon University. He has been a Research Staff Member since 2003, working on algorithms for automatic storage resource management.

Omer A. Zaki *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (omerzaki@us.ibm.com)*. Mr. Zaki is a member of the Advanced Storage Systems Group. He received a B.S. degree in computer science and mathematics from Angelo State University and an M.S. degree in computer sciences from the University of Wisconsin at Madison. Mr. Zaki's interests are in storage systems and databases.

Manuel Hernandez *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (manny@almaden.ibm.com)*. Mr. Hernandez provides electrical engineering prototype and automation solutions to the scientific community at the IBM Almaden Research Center and other IBM sites. He received a B.S.E.E. degree from California Polytechnic State University.

Kenneth R. Fernandez *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (kfernand@us.ibm.com)*. Mr. Fernandez is a mechanical designer with the Science and Technology Department at the IBM Almaden Research Center. He has been involved in prototyping and design of research and test equipment for computing and medical instrumentation since 1985. Mr. Fernandez was responsible for the mechanical design of the IceCube base platform and cooling systems.

Harald Huels *IBM Engineering & Technology Services, IBM Deutschland GmbH, Hechtsheimer Strasse 2, 55131 Mainz, Germany (huelsh@de.ibm.com)*. Mr. Huels studied communications and microprocessor technologies at the Universities of Giessen and Friedberg, Germany. He was the chief designer of the processor electronics of IceCube. Mr. Huels is currently responsible for the hardware architecture of PowerPC*- and Intel x86-based servers.

Heinz Lenk *Johannes Gutenberg Universität, Institut für Physik, Staudingerweg 7, 55099 Mainz, Germany (lenkh@uni-mainz.de)*. Mr. Lenk studied electronics at the Kaiserslautern Technical University, Germany. He joined IBM Mainz in 1977 as a test engineer, working on automated test systems for disk drive mechanics, electronics, and heads. Mr. Lenk retired from IBM in 2003 and is currently a development engineer at Johannes Gutenberg University.

Klaus Smolin *IBM Engineering & Technology Services, IBM Deutschland GmbH, Hechtsheimer Strasse 2, 55131 Mainz, Germany (smolin@de.ibm.com)*. Mr. Smolin studied electronics and computer science in Berlin. He joined IBM in 1994 and developed disk drive diagnostic applications, Linux device drivers, PCI add-in cards, and a PCI Express Interface for a new blade server.

Manfred Ries *IBM Engineering & Technology Services, IBM Deutschland GmbH, Hechtsheimer Strasse 2, 55131 Mainz, Germany (mries@de.ibm.com)*. Mr. Ries joined IBM in 1974 with a degree in mechanical engineering. He has worked in the design and modeling of mechanical devices, test equipment, design studies for throughput, and 3D solid modeling and design. His current interests are in finite element analysis for fluid flows and heat transfer simulations.

Carsten Goettert *IBM Engineering & Technology Services, IBM Deutschland GmbH, Hechtsheimer Strasse 2, 55131 Mainz, Germany (cgoetter@de.ibm.com)*. Mr. Goettert received a B.A. degree in engineering. He designed the power systems for IceCube and Cell Broadband Engine processor in 2003. His current interest is in highly efficient power system design.

Thomas Picunko *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (tpicunko@us.ibm.com)*. Mr. Picunko is a Senior Engineer in

the Central Scientific Services Department. He received an M.S.E.E. degree from New York University. He conducts engineering development projects for IBM internal customers. Mr. Picunko received an IBM Outstanding Technical Achievement Award for his work on a patterned wafer inspection system.

Barry J. Rubin *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (brubin@us.ibm.com)*. Dr. Rubin received a Ph.D. degree from the Polytechnic Institute of New York. He has worked on all aspects of electrical package analysis and is currently interested in gridding and hierarchical algorithms and techniques for accurate, robust calculations.

Howard Kahn *Virginia Polytechnic Institute, 323 Donaldson Brown Hall, Blacksburg, Virginia 24060 (howardkahn@gmail.com)*. Mr. Kahn graduated from the University of California at Berkeley with a B.S. degree in electrical engineering and is currently working on an M.S. degree at Virginia Polytechnic Institute. He participated in the IceCube project as an intern at the IBM Almaden Research Center.

Timothy Loo *University of California, 205 Cory Hall, Berkeley, California 94720 (timloo@gmail.com)*. Mr. Loo received a B.S. degree in 2005 and is pursuing an M.S. degree in electrical engineering and computer science at the University of California at Berkeley. As an intern at the IBM Almaden Research Center, Mr. Loo contributed to the research of the capacitive couplers and the construction of the IceCube bricks.