Perceiving the Dark Side of the Moon — Knowing When Scale-up Computing Makes Sense

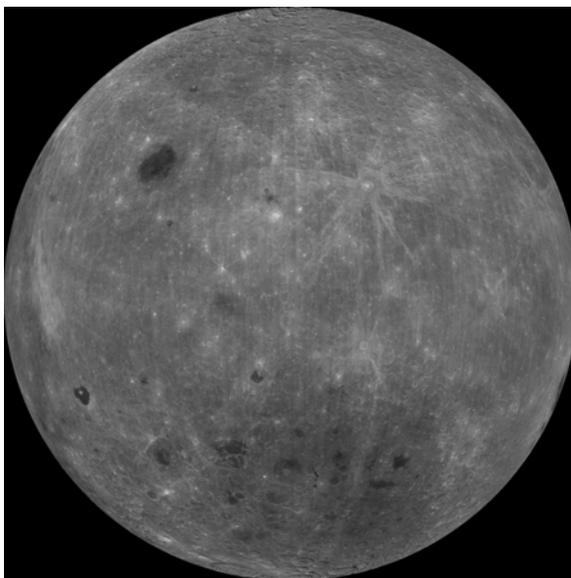Analysts: Mike Kahn and Bev Kahn

Management Summary

From Earth, we always see the same side of the Moon. While the Moon cycles through its phases, we see part or all of the same face, except when it is in eclipse (called a *New Moon*). Its face becomes familiar, even baked into our brains, as to what a moon should be. However, an equal amount of light shines on the other side of the Moon, the part we can't see, but because we can't see it, we have named it *the Dark Side*. In this case, *out-of-sight* is really *out-of-mind*. If we don't think about it, we begin to forget that it is there. You might say that a spotlight enhances the perception of reality and that **what isn't being illuminated for us to see can be ignored, forgotten, or even undiscovered**.

This issue of Clipper Notes sheds light on something that, for many, is unknown and in need of re-discovery. The subject is *scale-up computing,* **which today might be defined as** *a multi-processor architecture that manages shared server resources among many applications safely and scales "within the box".*[1] Hang on and don't disconnect just because this is beginning to sound like some outmoded legacy architecture that might not have even been taught (possibly, even way back) when you were in college. **What follows is a spotlight on the other side of computing – not really the dark side – but a richness that has grown out-of-sight and out-of-mind for many I.T. professionals.**

In the end, **a case will be made for** *scale-up servers* **as the most effective and most efficient approach for satisfying large-scale enterprise requirements for diverse, often interrelated applications with complex or integrated data sets, including databases, email, systems management, and** security. **This is about money and its data center equivalents: capital costs, energy and cooling costs, floorspace requirements, software licenses, and administrative and management overhead.** So, if you're not interested in exploring the possibility of completing certain workloads in a manner that is equal to or better than what you are doing now (in terms of satisfying Service Level Agreement (SLA) requirements and with a lower total cost of ownership (TCO)), then this is a good time to move onto something that you deem more important. Otherwise, read on for the details.

Exhibit 1 – The Unfamiliar Side of the Moon

Source: NASA

---

[1] As opposed to scaling by adding another server box.

Scale-Up Computing Explained

In a perfect world, we would all live within our means or, not likely, all have infinite resources. We would spend less than we earn. Our garages would be filled with our vehicles and not the flotsam and overflow from the many activities of our lives. Even better, in this perfect dream, we would be able to accurately anticipate our needs and adjust accordingly, an *on-demand lifestyle* of sorts. Well, the credit crisis demonstrates that many have not lived within their budgets. The plethora of neighborhood mini-storage sites shows that we often have more to store than we have room to keep at home. And, the dynamics of life tend to be anything but well-understood, well-planned, and well-managed. **You might say that we have become a generation that** *goes with the flow*.

Satisfying the Need for More Server Power

In the server world of the last decade, *going with the flow* meant adding more servers whenever they were needed. Really, what was being added was *server-processing power*, analogous in many ways to the *horsepower* of our vehicles' engines. This all used to be simple to understand because each of these servers was presumed to have one or two *processor*s and the horsepower (capacity to get work done) of the processor was largely determined by the frequency of the chip's clock and the amount of memory available. **We all knew, implicitly, that the next generation of servers would deliver more server-processing power, because the chips would go faster, and that there would be more memory available, all at about the same or a lower cost than the prior generation of servers.** In the "old days", we never had enough server processing power. **However, in an era of 16 cores (or more) per server[2] and a gigabyte of memory per core, we have – in theory – a lot more processing power in a single server than most folks can fathom**.

Architecturally, what has happened is that the server has again expanded to be able to do more than the preceding generation could do. Unfortunately, many or most operating systems and applications cannot take full advantage of the potential server processing power of an 8- or 16-core server (for a variety of reasons, but primarily because most business applications are *single-threaded*[3]). Thus,

it would be better for these single-threaded applications if we could have a single core with 16 times the power rather that 16 cores each with one-sixteenth the power, but the laws of physics and cost of energy prevent that from happening.[4]

What this means is that **the fundamental data center building block has changed from an** *ever-more-powerful single- or dual-processor server*, **where each processor has one core,** *to a power-per-core-restrained* (**i.e., with a very low increase in processing power per core**), *higher-core-count server*. Understanding the related implications is very important, especially with constrained capital, operaing, and energy budgets.

It's like your best supplier changing the size and functional capacities of a standard building block. We all recognize that this building block has been changing (improving) every six-to-nine months. A new rackmount server from a year ago likely is less performant that one that can be bought today. **Unfortunately, we tend to use this capacity growth as our favorite fudge factor; we count on moving higher-demand applications to the newest servers, to ensure that there is more headroom (excess capacity) available, for use when it might be needed.**

Dealing with Underutilization

This, of course, leads to the widespread existence of underutilized servers, which until recently, seemed to be acceptable because it simplified the management and attention required by the server administrators.[5] Unfortunately for the person paying for the servers, this caused many applications with modest growth to consume less-and-less of the total power of a *full server*. **In these harder economic times, this practice of planned underutilization is very wasteful.**

The confluence of improved server capacity and the woe of underutilization gave birth to widespread server consolidation via virtualization.[6] What this means is that a core can be

---

Some high-performance computing and database applications can be run in a parallel environment, with proper programming.

[4] Simply put, we can't manufacture circuits on silicon much more densely than is being done today, because there is too little space between the circuits (one or two atoms) and electrons tend to jump to an adjacent circuit when driven at high speeds. We can get some improvements by cooling the processors (so that they can run faster without thermally-induced errors, but we are not going to easily shorten the distance that electrons must travel without deploying new materials (other than CMOS).

[5] Corollary: too little headroom increases risks of failure.

6 For more background reading on server virtualization, see *Server Virtualization Made Real (Part 1 of a Multi-Part Series on Server Virtualization)* in the February 27, 2007, issue of **Clipper Notes**, available at http://www.clipper.com/research/TCG2007028.pdf.

---

[2] Four quad-core processors, for example. Recently announced servers based on Intel's 6-core "Dunnington" processor tie together 16 processors with six cores each, for a total of 96 cores.

[3] *Single threaded* means that the process runs in a serial fashion and that a subsequent part can't execute until all prior parts have completed. This is the opposite of *multi-threaded* applications, which can execute some or all of its sub-parts in a parallel fashion.

logically partitioned to look like a separate server but with only a fraction of the whole cores' server-processing power. This allows many applications that may have occupied separate servers (think of file and print servers, as a good example) to run in a single core at the same time (and even more, on multiple cores). Each virtual server has its own licensed instance of the operating system and continues to chug away doing what it did before, unaware that it is now working within a virtually-defined space called a *partition* (or *container*). Millions of older servers are being consolidated in this way, to run in a *fractional server space*, resulting in higher server utilization and lower energy consumption.

The Need for Something Bigger

Consider the following example. You have a commonplace fiberglass boat with a 10-horsepower outboard engine and you in the business of delivering groceries to residents of island communities off the mainland. A boat's maximum speed is limited by the length of the hull, at least until it starts skipping across the water (like a jet ski), and the power of the propulsion system (i.e., the outboard engine). Let's say that it can travel five miles per hour with only one occupant and no cargo, but you want to go faster and carry a greater load.

You have two choices, assuming that the boat can handle the extra weight. You can buy a larger outboard engine; say, one that puts out 20 HP. Or, you can get a second 10 HP engine and run both of them together. You can extend this thinking until you reach a point that no matter how much total horsepower you have onboard, the boat won't carry any more or go any faster. In fact, it might sink under the shear weight of the infrastructure, including the fuel needed to power it.[7] What you really need is a *bigger boat*, capable of carrying more weight (motors, fuel, and cargo) and maybe even able to go faster. Or, maybe you need additional identical boat(s), each with its own driver.

If you had a fleet of 10 boats, you could have each of them deliver to a different route simultaneously, but each would require a high-cost driver. You would have great flexibility to make independent deliveries, but there would be no opportunity to optimize the deliveries by location, since only a little cargo could be carried in each boat.

What you really need is a bigger boat (capable of carrying more cargo, i.e., a greater load) that also minimizes the collective crew required. Yes, a bigger boat will cost more to buy and operate, and will

be of a more complex design than your smaller boat, but it will be safer to operate[8] and its business efficiency will be higher[9].

Introducing Scale-Up Computing

This nautical example portrays the story of scale-up computing architectures. Here's how.

1. There is a real limit to how much server processing power can be delivered by a single core.
2. There is a limit to how many independent cores can be put in a simple server before the weight of the infrastructure (or the price of the server or the energy required) begins to deliver performance (or cost) degradation.[10]
3. Most importantly, there are some situations where you just need a "singular" higher-performance processing vehicle than can be created within the limits of a single core. In IT, we might call this a need for *scalable capacity*. **In hardware terms, what you need is multiple cores (think "many" not just a "handful") that work like a single core**, i.e., you need a "bigger boat" designed for carrying a greater load with improved efficiency.
4. You would benefit from this singular, completely-shared resource, which in this case is about the large amount of memory that can be shared (like a common cargo area).
5. Furthermore, what you would really like is a vehicle that could deploy a varying number of cores (engines), so that you could deploy and power only what you needed to meet the current requirements, with other engines (additional processing power) available to be added, if your workload demand spikes unexpectedly.

Does this mean that there is no place for smaller, independent computers, working either separately or in some kind of coordinated manner? *Absolutely not*. **This just means that there are some applications that will benefit from the ability to** *scale up* **within a** *shared delivery vehicle*.

Two Kinds of "Big" Applications

So, what is a *big application*? From the data center's point of view, it's something that consumes a lot of resources. Smaller applications are being consolidated (through virtualization) because they

---

[7] You might say that this is what is happening today to those enterprises with thousands of rack-mounted servers.

[8] Larger boats are better able to weather a storm.

[9] Because the cost to deliver a hundred pounds of cargo per mile travelled will be less. Thus, the heftier design point of the bigger boat, the potential for delivery route optimizations, and the reduced crew required, are economically beneficial for the work to be done, in comparison to having many smaller boats.

[10] This is true when you have more work than can be done on a single core. If the workload is small, then server virtualization can be used to create many fractional partitions (of a core), and the TCO should be less because of the virtualization.

are not resource hogs. It's usually the big applications that stress the data center's staff (to keep them running without incident) and that cause the enterprise many problems when they are not available or performing very slowly. The one that comes first to most of us is *email*. If you have a large number of employees who can't reference, send, and receive email, then you have enterprise paralysis of the first order. If a bank branch can't use its teller systems, it will quickly become a grumpy place. If a financial planner can't run simulations of different portfolio strategies, unproductivity abounds. Each of these is a big application because they are either data intensive and/or processing intensive and shared by many users simultaneously.

These big applications are different from the word processing that we do on our laptops or the Yahoo or Google mail that we check from that same machine. The first is a self-contained application (that runs independently of other applications and users). And, the second is just a browser window serving as a thin client (with the processing work being done somewhere else). For the purpose of this paper, let's consider two kinds of big applications.

- *Serial applications*, which tend to run in a strictly *linear* fashion, sort of like the transactions on your bank account. The processes must be done in order and each process depends on the results of the prior process(es) to act properly. You might see the application as a single-server queue (like the line at a grocery store, where there is a cashier and register for each line of customers). When there are too many customers waiting, the store manager might open additional registers and the lines redistribute themselves to take advantage of the newly available resources.

- *Parallelizable applications*, which tend to execute in totally isolated parts and where the results of the parts may be brought together (re-aggregated) into a collected result.

Some new applications are being designed to run in a multi-core environment, i.e., using many cores.[11] In most cases, these applications manage the division of work among the many cores. This kind of parallel operating environment can work well for some, largely *compute-intensive*[12] applications, like scientific computation and business

modeling.

However, and a very big "however" at that, **most business applications that run on *Windows* and *Linux* were designed to run serially in a single thread (or maybe a few). Thus, most applications are limited to using the server power of a single core (or a thinner virtualized slice thereof).**[13] **If you can see this as being true in your data center, then you know that there are applications that need more server processing power than the increase of power per core has been delivering.** Exactly how are these needs being satisfied? What kinds of applications tend to require more processing than can be done on a single core?

The simple answer: *big applications*. OK, that's not a very useful answer. How about *operating environments that manage their own data storage space and I/O*. Clearly, this includes server operating systems (like Windows and Linux), whole-server applications (like Microsoft *Exchange*), and database management systems (like Microsoft *SQL Server* or *Oracle*). **In these cases, the operating software assumes the responsibility for managing the resources provided by the server (processing power, memory, I/O, etc.).** Now, we are getting closer to the topic of the day!

Therefore, there are really important applications that might need and might be able to take advantage of the increased server-processing power that multiple cores working together can provide. There are two ways to think about this.

Putting Multiple Cores to Work

First is the way described above, where the operating environment (or a sophisticated application running therein) knows about and understands how to use multiple physical execution vehicles (cores/processors).

Second, and also very important, is when the server hides all of this complexity and *scales up* the available cores (i.e., aggregates processing power transparently) and allows managed access to shared memory. When this is done, it is called *SMP* (for *Symmetric Multi-Processing*). **SMP allows many cores of the server (think big, as high as 96 cores now, with more to come) to**

---

[11] Like a fleet of separate delivery boats, each working independently of the others.

[12] High-performance performance applications tend to be "computing intensive" while most other applications (database, transaction processing, web access, etc.) tend to be I/O (input/output) intensive.

---

[13] Most laptops and desktops today are dual-core computers. For you to get any throughput advantage from that second core, you need independent applications, i.e., those that are isolated from each other. Thus, a second core can download your email while you type into a word processing application on the first core, which might net an improvement over a single-core computer. However, the second core will do little to help you create a PDF from a word processing document, because the conversion software only can work through the process serially.

look as if they were just one workspace or maybe a few, securely isolated workspaces. This is achieved through software, primarily, but works better when it is hardware-enabled. **Therefore, if you need the power of *many* cores to drive your big application, you can do that quite easily with a *scale-up server*.**

This even gets better. Those of us who focus on delivering infrastructure always live in fear of the unexpectedly "big moment" or "holiday season", where demand might be many times what happens normally. Think about handling the immediate wave of responses to an effective *Super Bowl* commercial. On a lesser scale, this might be an end-of-quarter closing of the financial accounting system. These are times when more server-processing power will be needed.

Now, either you can have it sitting idle most of the time, to be ready for that just-in-case overload, or you can increase – dynamically – the number of aggregated cores for the involved application in anticipation of or in response to a rapid escalation of demand. You might be using the otherwise idle cores for development work, data marts, etc., which you can scale back when more cores are needed for a business-critical application. **This dynamic scaling can be done quickly (and safely) on advanced scale-up (SMP) systems.** Think of each independent domain (an aggregation of many cores of processing capability) as an accordion, whose resources (cores) can be inflated (or expanded, i.e., more capacity added) or deflated (no-longer-needed cores returned for others to use), when required, according to policies and conditions that were defined in advance – or by on-the-fly changes by system administrators. Some call this *On Demand* or *Dynamic Computing*. **Whatever you call it, this is where you want to be.**

The Benefits of Sharing

Yet there still is more! SMP computers have shared memory, which means that all applications can access all of the memory installed in the scale-up server.[14] Sharing memory has a number of important operational merits.

1. ***Data in memory can be shared by many applications.***[15] Rather than going to another system

(node) over the network to get access to shared data (an overhead-burdened process), **with shared memory in an SMP server, the data access is made at memory reading speeds (much faster) and the work can get done faster.**

2. *Applications in memory can also be shared.* **A well-designed application can exist in shared memory and be *re-entrant*,** allowing potentially hundreds of different users to enter and leave the same instance of code (again, with protections for securing each user from another's data. Thus, you might have only one copy of the DBMS (or operating system) in memory rather than one for each user or application. This sharing cuts out the wasteful, redundant use of memory (still a scarce resource) as a holding place for an application's code. This might also affect the application's license costs, usually for the better.

Conclusion

Thus, if you can share the operating environment, the application, and the DBMS, and still protect data and restrict users to what they are entitled, then you begin to see why scalable SMP architectures might actually be more economical and more efficient than *scale-out* (grid-like, asymmetric) architectures.[16] While scale-out architectures inherently have increased hardware costs over a grid of nodes, being able to enhance performance of big applications, to resize the partitions dynamically, and often to significantly reduce software license fees and administrative overhead makes it cost effective.

Big, business-critical applications do exist and usually will perform better and operate more economically in scale-up server environments. You must understand these facts in order to choose the right server platforms, especially for your big applications. **Boldly consider your critical requirements and the benefits of the scale-up infrastructure alternative!**

---

[14] Of course, this is supervised to limit each application and user's access to what is entitled.

[15] This can be a powerful accelerator in solutions based on applications employing a Service-Oriented Architecture (SOA), where data and applications are assembled on the fly (often called *mashups*. If the component pieces are in shared memory then they can be access (and mashed up) more quickly. For more on SOA, see the September 23, 2008, issue of *Clipper Notes* entitled *The Role of Governance and Service-Oriented Architectures*, at http://www.clipper.com/research/TCG2008047.pdf.

[16] Remember, in contrast to scale-out architectures, where additional server nodes are added endlessly, SMP servers are said to *scale up*, because they build on the same base of infrastructure as they are scaled (by adding more processors, memory, networking connections, etc.).

About The Clipper Group, Inc.

The Clipper Group, Inc., is an independent consulting firm specializing in acquisition decisions and strategic advice regarding complex, enterprise-class information technologies. Our team of industry professionals averages more than 25 years of real-world experience. A team of staff consultants augments our capabilities, with significant experience across a broad spectrum of applications and environments.

➢ *The Clipper Group can be reached at 781-235-0085 and found on the web at www.clipper.com.*

About the Authors

Mike Kahn is Managing Director and a cofounder of The Clipper Group. Mr. Kahn is a veteran of the computer industry, having spent more nearly four decades working on information technology, spending the last 15 years at Clipper. For the vendor community, Mr. Kahn specializes on strategic marketing issues, especially for new and costly technologies and services, competitive analysis, and sales support. For the end-user community, he focuses on mission-critical information management decisions. Prior positions held by Mr. Kahn include: at International Data Corporation - Director of the Competitive Resource Center, Director of Consulting for the Software Research Group, and Director of the Systems Integration Program; President of Power Factor Corporation, a Boston-based electronics firm; at Honeywell Bull - Director of International Marketing and Support; at Honeywell Information Systems - Director of Marketing and Director of Strategy, Technology and Research; with Arthur D. Little, Inc. - a consultant specializing in database management systems and information resource management; and, for Intel Corporation, Mr. Kahn served in a variety of field and home office marketing management positions. Earlier, he founded and managed PRISM Associates of Ann Arbor, Michigan, a systems consulting firm specializing in data management products and applications. Mr. Kahn also managed a relational DBMS development group at The University of Michigan where he earned B.S.E. and M.S.E. degrees in industrial engineering.

➢ *Reach Mike Kahn via e-mail at Mike.Kahn@clipper.com or via phone at (781) 235-0085 Ext. 121. (Please dial "121" when you hear the automated attendant.)*

Beverly K. Kahn, Ph.D. is Research Director for The Clipper Group. Dr. Kahn specializes in the quality of information and methods for improvement. She is also Chair, Information Systems and Operations Management Department, Sawyer Business School, Suffolk University, Boston, where she has been a professor since the mid-1980s. Previously, she was an Assistant Professor at Boston University. Dr. Kahn received B.A., M.S., and Ph.D. degrees from the University of Michigan.

➢ *Reach Bev Kahn via e-mail at Bev.Kahn@clipper.com or via phone at (617) 573-8642*

Regarding Trademarks and Service Marks

The Clipper Group Navigator, **The Clipper Group Explorer**, **The Clipper Group Observer**, **The Clipper Group** *Captain's Log,* and *"clipper.com"* are trademarks of The Clipper Group, Inc., and the clipper ship drawings, *"Navigating Information Technology Horizons"*, and *"teraproductivity"* are service marks of The Clipper Group, Inc. The Clipper Group, Inc., reserves all rights regarding its trademarks and service marks. All other trademarks, etc., belong to their respective owners.

Disclosure

Officers and/or employees of The Clipper Group may own as individuals, directly or indirectly, shares in one or more companies discussed in this bulletin. Company policy prohibits any officer or employee from holding more than one percent of the outstanding shares of any company covered by The Clipper Group. The Clipper Group, Inc., has no such equity holdings.

Regarding the Information in this Issue

The Clipper Group believes the information included in this report to be accurate. Data has been received from a variety of sources, which we believe to be reliable, including manufacturers, distributors, or users of the products discussed herein. The Clipper Group, Inc., cannot be held responsible for any consequential damages resulting from the application of information or opinions contained in this report.