Thirty-Two-Bit Fat Clients Have Hit the Wall: Consequences for TRIP™

Steve Langer

In its 2003 Annual meeting, the Society of Computer Applications in Radiology launched the Transforming the Radiologist Interpretation Process (TRIP™) initiative. As described at the time, TRIP™ is intended to assist radiologists with the tsunami of exploding data sets that they are increasingly expected to interpret. TRIP™ initially focused on three high level strategic objectives:

1. Improving efficiency of interpretation of large data sets,
2. Improving the timeliness and effectiveness of communication, and
3. Decreasing medical errors.

However, there is a fundamental consideration that must presage the above concerns. Namely, will the computer architectures used even be capable of swallowing the data torrent that is pointed at them?

KEY WORDS: High performance computing, software design, CAD

INTRODUCTION

In 2003, Society for Computer applications in Radiology launched a new initiative at the annual meeting. The Transforming the Radiological Interpretation Process (TRIP™) was forward thinking and correctly identified that the rise of multi-detector computed tomography (CT) was just the first assault in the rising radiologic data tide (some would say tsunami)¹. In the first TRIP™ conference (Bethesda MD, 2003), posters and presentations covered the following: human perception, image processing and computer-aided diagnosis (CAD), data visualization, graphical user interface and navigation, databases and systems integration (RIS, PACS, speech recognition), and finally systems evaluation and validation². In the database and integration section, session leader Steve Horii pondered if databases would be able to

scale to the size and speed that would be required of them. This concern turned out to be presciently close to the concern of this paper.

A follow-up review article provides an excellent summary of the literature as of 2004 as it pertains to the points of interest in the initiative³. A second TRIP™ conference was held with another round of inventive presentations spanning CAD, perception, decision support integration, etc. Progress was being made in several areas, yet still a fundamental point was overlooked.

EXAM SIZE TRENDS

To understand the motivation for this work, it is instructive to look at the size trends for CT at our institution. Our clinical fleet has approximately 23 CTs. Until mid-2005, the fleet consisted of a mix of single-, four-, and eight-slice units. From that point forward, the fleet has been migrating toward all 64-slice units; currently, that migration is now over 50% complete. As a consequence of the new capabilities, exam protocols have changed, and exam slice counts have exploded in response as shown in Table 1.

The majority of the slice count surge is not from exam volume growth, but rather from slice count

From the Mayo Clinic, 200 First Street SW, Rochester, MN, 55904, USA.

Correspondence to: Steve Langer, Mayo Clinic, 200 First Street SW, Rochester, MN, 55904, USA; e-mail: langer.steve@mayo.edu

Copyright © 2009 by Society for Imaging Informatics in Medicine

Online publication 12 February 2009

doi: 10.1007/s10278-009-9186-8

Table 1. CT Growth Rate in Total CT Slices Stored Per Year

Year	CT image count
1997	2,760,759
1998	3,485,113
1999	4,520,836
2000	5,544,986
2001	7,307,608
2002	9,746,599
2003	13,011,629
2004	17,565,699
2005	26,368,808
2006	46,516,823
2007	63,911,019

increases per exam from the new protocols. Formerly, the largest CT exams in our practice were typically 2,000 slices. Now, the standard trauma chest–abdomen–pelvis protocol results in exams of 5,000 slices. It is not unusual for a normal chest–abdomen–pelvis to have 3,000 slices, and given two or three historical comparisons, the slice count expected to be pulled and viewed into a workstation is 12,000 $512 \times 512 \times 2$ -byte slices. When one considers that a single CT slice is 0.5 MB, the situation above translates to 6 GB of memory that is required simply to hold the images for display on a workstation.

As mentioned earlier, multi-detector CT was just the first wave. Functional magnetic resonance (MR) exams in our practice can now exceed 15,000 slices of various matrix sizes. Often, these are long-term follow-up patients with multiple comparisons, including positron emission tomography–CT. The result is that workstations are expected to digest data sets whose size grow without limit. The following table shows the typical image size (in bytes) of common modalities (note the changes of some modalities over time; Table 2).

Finally, one can combine the evolving image size changes with the impact changing slice counts

Table 2. Modality Image Sizes

Modality	Matrix size/slice	MB/slice
CT	$512 \times 512 \times 2$	0.5
New MRs	$512 \times 512 \times 2$	0.5
Old MRs	$256 \times 256 \times 2$	0.125
Digital Fluoro	$1024 \times 1024 \times 2$	2
CR/DR	Variable	8

Table 3. Changes in Overall Exam Size from 2002 to 2008

Modality	02 Slices/ exam	02 MB/ exam	08 Slices/ exam	08 MB/ exam	Comment
CT	300	150	3,000	1,500	
MR	300	37.5	1,200	600	New MRs
Digital Fluro	150	300	900	1,800	
CR/DR	3	24	80	640	Tomo DR

have on exam size and see the memory load that is expected to be born by the PACS workstation. Then, consider there may be two, three, or more comparison exams (Table 3).

COMPUTER ARCHITECTURES

At core, computers consist of a central processing unit (CPU) connected to volatile memory (RAM, random access memory), static memory (disks), networks, video, keyboard, and mouse. One of the key features used to describe a CPU is the bit width of its internal memory registers. These are the memory cells that add, subtract, and perform all other chip operations. Often, the register width is the same as the width the chip presents to the memory bus. The old Intel 386, 486, and Pentiums were 32-bit chips with a 32 conductor path to interface with RAM (Intel, Santa Clara CA). This 32-bit address space sets an upper limit to the amount of RAM the chip can access. A simple example will help to illustrate. One bit can encode two values: 0 or 1. Two bits can encode 2×2 values: 0, 1, 2, and 3. The pattern is obvious: 32 bits is $2 \times 2 \times \dots \times 2$ 32 times or 2 raised to the 32nd power. This turns out to be 4,294,967,296 or roughly 4.3 billion values that can be stored and uniquely addressed.

This sounds like a large number. Except...from the above example, we need six billion bytes to store 12,000 CT slices. Six is obviously bigger than four.

In actual practice, the situation is worse. Not all the workstation's RAM can be reserved for just holding images. Some is used by the computer operating system. More is used for the PACS, speech recognition, RIS work list and decision support applications. At our institution, we built

our PACS workstations with 8 GB of RAM (yes, more than a 32-bit chip can use, we will see why later) and we observe an interesting pattern in memory usage. With PACS, speech recognition and the work list loaded, the total free RAM available is 1.7 GB on Windows XP-32 (Windows, Microsoft Corporation, Redmond WA, USA). That is 3,400 CT slices—total—for all unread and comparison exams.

All is not lost. For some time, Intel and other chip makers have seen this need coming, and since about 2004, 64-bit chips have been available for common desktop computers⁴. For reference, a chip with a 64-bit path to RAM can access 18,446,744,073,709,551,616 or 18 million TB of memory. In fact, many 64-bit chips do not fully exploit this wide a path to RAM, such as the AMD Opteron (AMD, Sunnyvale, CA, USA). An Opteron “only” uses 48-bit addressing, which is still 64,000 times more memory than the 32-bit chip can access. However, for this memory to be available, the PACS vendors have to recompile their applications to fully exploit the 64-bit operating system. In fact, this is a move we anticipated from our PACS vendor (recall the excess RAM used in the workstation build from above). We migrated our PACS workstation fleet to Windows XP64. However, the PACS vendor has declined to recompile their application, so the anticipated 64-bit benefit is unrealized.

SOFTWARE ARCHITECTURE

We have seen there is a “hard stop” to the capability of a 32-bit PACS workstation to be able to consume and display large data sets. We have also seen there is a potential to avoid this stop by migrating to 64-bit hardware. However, this can be expensive for a site if they did not plan for it and “future proof” their workstation hardware purchase by investing in 64-bit chips and memory at purchase time. Further, the vendor has to be willing to recompile their PACS application to exploit the new memory model.

Is there any way to extend the useful life of 32-bit workstations while still enjoying the ability to view increasingly larger data sets? It turns out there is, by re-architecting the viewing application.

One often hears the terms “fat-client” or “thin-client” at SIIM or other industry conventions.

What exactly do these terms mean? The term “fat-client” is fairly straight forward and refers to a software application that lives on the desktop computer, relies on the desktop operating system for networking and other services, has total access to the local file system, and has to pull all the data it intends to view to the local workstation before it can be shown.

“Thin-client” is a more nebulous term. It is often misapplied to fat clients that a user acquires from a web site, pulls down, and installs locally. To a large extent, a defining characteristic of a true thin client is that it *is isolated* from direct contact with the local operating system and instead has its service requirements met via communication with a remote networked server over web protocols^{5,6}. Put another way, if installing the client invokes the Microsoft Windows Installer, it is a thick client. Most true thin clients today are built on either .NET or J2EE (Java 2 Enterprise Edition) web programming frameworks (.NET, Microsoft Corporation; J2EE, Sun Microsystems, Santa Clara CA, USA). These frameworks standardize the service interfaces on the server to any web client. A critical point to understand is this: Does the application need to acquire and hold all study images locally before they can be viewed? If it does, it may be a thin client, but it can still exhaust the workstation’s RAM. What is needed is a “thin-client” with a “thin-RAM-footprint”.

Many individuals are familiar with viewing video in a web browser; in fact, entire TV shows can be viewed this way⁷. Depending upon the compression ratio, a digital recording of an hour of TV video (non-high definition) is typically about 2.5 GB. At typical download speeds of 1 Mbps, it would take about as much as 333 min to bring the video into one’s workstation to view. In other words, if the compression is slight and one has a 1-Mbps link, one would wait about 5.5 h to view an hour show. The fact that the real life experience is unlike this is because the video viewer does additional compression and also shows the images as they arrive over the network. Such a method is called “streaming,” as opposed to the store and hold model of most PACS views^{8,9}. Because the images are shown as they arrive and then typically discarded, both memory and waiting requirements are vastly reduced. In fact, we realized the value of this approach at our institution several years ago when developing our own next generation clinical

viewer. The clinician viewer is available on over 20,000 clinical workstations; the hardware ranges from machines with only 0.5 GB of RAM to over 2 GB. However, since the viewer is programmed in a thin-client streaming manner, CTs of over 5,000 slices are viewable across the fleet.

As we have seen, there are two benefits to streaming: it reduces the memory footprint on the local workstation and as an adjunct reduces the time, the user has to wait to view the first image. What is perhaps a bit less obvious is that streaming makes viewing *any particular image* faster. If the PACS user interface permits the user to randomly move among an image stack (via a slider control for instance), the server can interrupt whatever it is currently sending and jump to the slices near that relative offset in the series. What it does not improve is any operation performed on the workstation that *requires the entire series* to be present (i.e., 3D reconstruction) or multiple series to be fully present (i.e., CAD, fusion, or change detection). Of course, if such operations are server-based and only the results are sent to the workstation, the foregoing limits are mitigated.

CONCLUSIONS

We have seen the limits of 32-bit architectures and the potential to mitigate those limits with 64-bit computing. However, we have also seen the tepid embrace of 64-bit computing among the PACS vendors. There is another path, but it

obsoletes the design of the vast majority of currently installed PACS software. This path will require many customers to perform upgrades to permit their PACS to scale to the next level.

The era of 32-bit PACS “fat-clients” or “thin-clients with fat RAM foot-prints” has come to an end. If they are not already, PACS vendors will have to fundamentally re-architect their applications to survive in the TRIP™ era. This is a fundamental prerequisite before other TRIP™ concerns can be effectively addressed.

REFERENCES

1. Morin RL. <http://www.siiim2009.org/trip/html/TRIPed2.html>. Last viewed October 2008
2. Andriole KP, Morin RL: Transforming medical imaging: the first SCAR TRIP™ conference a position paper from the SCAR TRIP™ subcommittee of the SCAR research and development committee. JDI 19(1):6–16, 2006
3. Andriole KP et al. Addressing the coming radiology crisis—the Society for Computer Applications in Radiology Transforming the Radiological Interpretation Process (TRIP) initiative. JDI 17 (4):235–243, 2004
4. Bit processor timeline. http://en.wikipedia.org/wiki/64-bit#64-bit_processor_timeline. Last viewed October 2008
5. Niemeyer P, Knudsen J. Learning Java: 2nd edn. O'Reilly and Associates, 2002. ISBN 0-596-00285-8
6. McLaughlin B. Head Rush AJAX. O'Reilly and Associates, 2006. ISBN 0-596-10225-9
7. ABC Video. <http://abc.go.com>. Last accessed 30 December 2008
8. deCarmo L. *Core Java Media Framework*. Prentice Hall, 1999, ISBN 0-13-011519-3
9. Gordon R, Talley S *Essential JMF: Java Multimedia Framework*. Prentice Hall, ISBN 0-13-080104-6