Red Hat Enterprise MRG – Realtime Whitepaper

1 Executive Summary

Many enterprise workloads have extremely demanding requirements for determinism – to ensure predictable response times at the 20 microsecond (µs) latency level.[1] The difficulty has been that until now, all realtime operating systems have been niche offerings – absent of any COTS (Commercial Off The Shelf) products. Red Hat® is uniquely positioned to provide these capabilities due to its proven track-record of development and mainstream accepted **Linux kernel realtime enhancements**. In response to strong customer demand, Red Hat is productizing its realtime kernel in an offering called ***Red Hat Enterprise MRG***, comprised of high speed messaging, realtime kernel, and grid cluster scheduler.

Kernel level enhancements form the foundation of Red Hat's commercial offering. Consider that few computers today function completely independently. Rather, they are typically connected in a distributed environment. Red Hat's approach to a realtime solution stack extends beyond the kernelunderpinnings, to include:

- **High speed messaging** – via an open source AMQP messaging middleware offering from Red Hat, referred to as ***MRG Messaging***.
- Grid scheduler - enables enterprises and research organizations to bring the power of distributed computing across their entire infrastructure to tackle the largest computational problems in a highly efficient and effective manner. Referred to as ***MRG Grid***.
- **Realtime Java** – via strong partnership with IBM

This combination of engineering leadership, award winning Red Hat Enterprise Linux® distribution, open source messaging middleware, and established partnerships yield the most comprehensive realtime deployment platform.

This whitepaper describes the ***MRG Realtime*** capabilities being productized on a Red Hat Enterprise Linux 5.1 foundation. It details the implementation and productization strategy aimed at meeting the

1 20 µs is the time interval between when a trigger occurs (such as an interrupt or timer expiration) and when the scheduler runs the highest priority pending process running on realtime certified hardware.

most demanding customer workloads while at the same time preserving application compatibility.

2 The need for enterprise realtime

Historically the concept of realtime has been most applicable to embedded monitoring devices and control equipment. Today there is a growing class of enterprise applications which have demanding time constraints which are unable to be satisfied with general purpose operating systems. This often necessitates utilizing non-standard operating systems for certain workloads. These niche operating systems put an inordinate strain on enterprise IT organizations to acquire skills in system management and application porting to an obscure environment.

Red Hat has risen to the challenge to provide industry leading realtime capabilities in a commercial off the shelf operating system (COTS) - the Red Hat Enterprise Linux product family under the name MRG Realtime. By integrating realtime, high performance messaging middleware capabilities and grid scheduler into the most popular enterprise Linux distribution (Red Hat Enterprise Linux 5), IT organizations do not require retraining and a broad array of 3rd party applications are at your disposal.

2.1 Example realtime enterprise workloads

The standard Red Hat Enterprise Linux releases provide high performance to meet the requirements of the majority of enterprise workloads. For example, Red Hat Enterprise Linux is routinely used in world record breaking database benchmarks such as TPC-C[2], and TPC-H. Similarly the performance for general purpose web and file serving leads the pack. These workloads of file-server, database and web server are characterized by the need for *high throughput*.

There are other classes of applications where overall raw throughput alone is insufficient. In many enterprise workloads, *determinism – predictability in response time, even when the system is under heavy load* – is the paramount objective. This class of applications necessitate realtime capabilities. The following are representative examples of such scenarios:

- ***Financial services industry*** - here, time is money - in a literal sense. Guaranteed response time is of the essence. For example, it is not adequate if out of 100,000 transactions per minute if 99,999 complete in 2 ms, and 1 completes in 20ms. While that is statistically a low number, it represents one lost trade per minute. Recently introduced SEC regulations can result in severe fines if inconsistent trading times occur (as it hints of favoritism) – hence determinism is more important than ever.[3] The increasing use of algorithmic trading is raising the requirements for low latency

2 http://www.tpc.org/

3 Example government regulations concerning "best execution" and transparency include the US Securities and Exchange Commission *Regulation National Market System* (**RegNMS**), and the European Union's *Markets in Financial Institutions*

and high performance. Trading firms are admittedly engaged in a *low latency arms race*.

- ***Network devices*** - many TELCO and network based services (including deployment in traditional embedded devices such as bridges and routers) are extremely sensitive to response time. Without high predictability of response time in running their network control applications, there would be an unacceptably high rate of packet loss and general degradation of service.
- ***Command and control*** - Many military and industrial control applications require deterministic response time for control applications. Here it is critical that the priority of the control applications does not get superseded by non-essential processes and operating system tasks. The ability to fine-tune the priority of both system and application processes with high predictability is essential.
- ***Realtime java applications*** - there is a growing class of applications utilizing RTSJ (RealTime Standard Java) capabilities[4]. Any RTSJ compliant JVM has stringent requirements on the operating system for low latency and priority inversion avoidance capabilities. It is not uncommon for Java applications to have huge numbers of threads, ie 10's of thousands. Such workloads require a fully certified combination of an RTSJ compliant JVM and underlying operating system to scale accordingly in a deterministic manner.

3 Low latency and predictability defined

The previously mentioned realtime workloads have 2 primary requirements:

- **Predictability in response time** – ensuring *consistency* in response times
- **Deterministic upper bound on latency** – ensuring that processing will return correct results within a constrained period of time; every time – referred to as *guaranteed response time*

Predictability in response time primarily amounts to ensuring that the highest priority processes are run first. Traditional Linux offerings suffer from a problem known as Priority Inversion (PI). PI can occur when a high priority process requires a resource (ie lock) held by a lower priority process, hence the high priority process is blocked. In the meantime the lower priority process may yield to a medium priority process. This medium priority process could run for a long time, effectively stalling out the higher priority process. To avoid this situation our realtime product temporarily boosts the low priority process - effectively to allow it to complete its work and release the resource to the high priority process. This is just one of many examples.

In addition to considering priorities of user application processes, there are also a wide variety of

Directive (**MiFID**).

4 http://www.rtsj.org/

operating system threads, including interrupt servicing. Depending on the application workload, in realtime scenarios it is necessary to be able to prioritize certain application processes above some operating system kernel services. For example, consider the case of an industrial control application running on a 4 CPU system. For optimal performance it may be configured to have the control application running exclusively on 2 processors, have all network interrupt handling performed on the 3rd CPU, and the remainder of general operating system tasks and non-time-critical application processes constrained to the 4th CPU.

Another traditional sore-spot in Linux preventing dependable determinism is that there are numerous extremely lengthy kernel codepaths. Our realtime engineers have been methodically breaking down unnecessary serialization through techniques such as deferring the majority of interrupt handling to lower priority kernel threads. We have also improved contention points in multiprocessor locking - yielding scalability benefits to general purpose Red Hat Enterprise Linux and realtime workloads alike. The traditional measure of deterministic upper bounds on latency consists of the time it takes from when an event triggers (such as a device interrupt, timer interrupt) to the scheduler yielding to the corresponding highest priority application process – referred to as **scheduler latency**. In the Red Hat Enterprise Linux-RT variant, this worst-case scheduling latency is under 25 µs (typically under 20 µs on validated hardware[5]). The **jitter** (standard deviation from the mean value) is in the 2-4 µs range. This bounding of system overhead is particularly important for realtime application workloads as it makes the overall response times (including system overhead and application execution) more predictable.

Additionally, the precision of timer granularity has been improved to the nanosecond resolution (provided the underlying hardware is capable). For example, in standard Red Hat Enterprise Linux 5 the most precise sleep duration is 2 ms. In contrast,MRG Realtime has 10 µs capability (hardware dependent).

4 Red Hat - the Linux realtime innovation leader

Deploying Linux in realtime environments is not a new phenomenon. Companies have been doing this for years. The problem to date is that, none of the commercial offerings have been mainstream Linux - all prior realtime offerings have been the exclusive province of custom, proprietary, niche offerings. History has proven that niche offerings do not stand the test of time and expense constraints. The profound industry trend is to mainstream COTS offerings.

In the past, there have been several attempts at integrating realtime capabilities in the mainline Linux codebase. All such integration attempts consisted of proposing colossal change sets - typically on the

5 Scheduling latencies are very hardware dependent. For example, the same kernel consistently yields 15 µs on a newer system and 150 µs on an older system (circa 2005).

order of 10's of thousands of lines of code - which in the Linux community are dubbed "***patch bombs***". The fatal flaws with the "patch bomb" approach include the following:

- It is virtually impossible for the broad community to understand and approve due to their enormity – termed *not consumable*.
- They are typically *short-sighted*, focusing on a niche market - and detrimental to general purpose operation. In fact, there were so many failed attempts with realtime patch bombs that it was decreed that it would *never* become mainstreamed in Linux.

Among all the companies with strong intentions for realtime Linux capabilities, Red Hat has taken a unique approach. Rather than taking a typically proprietary approach of developing realtime capabilities in-house and later patch-bombing the Linux kernel community upon completion, Red Hat's strategy from day 1 has been to establish a patient multi-year incremental roadmap. We have broken down the problem into first getting in enabling capabilities and infrastructure cleanup. With each new mainstream kernel[6] version we have continued the progression. Red Hat has been doing all of this development in a fully open and inclusive manner. To avoid the stigma of prior companies failed realtime attempts, we have methodically proven that all of the enablers which we have successfully integrated upstream are demonstrated to be beneficial to general purpose computing. In this way we have raised the overall bar for Linux in general. You could say that Red Hat's approach to mainstreaming realtime Linux can be summarized by the following parable:

- *Question: How do you eat an elephant?*
- *Answer: One bite at a time.*

Red Hat's open, inclusive, upstream focused strategy to delivering realtime capabilities is completely consistent with the well proven Red Hat Enterprise Linux productization methodology which has made it the industry leading Linux distribution. The efforts of Red Hat's engineering staff have made us the upstream engineering project leaders and implementers of realtime capabilities. There we have formed a developer community focusing on realtime features and invited broad participation[7]. An increasing number of Linux vendors are recognizing the success of this initiative and starting to jump on the bandwagon – often by abandoning their prior non-mainstream alternative.

4.1 Demonstrated mainstream realtime results

The following table highlights many of the realtime related features and infrastructure which Red Hat

6 *Mainstream kernel* – refers to the primary open source kernel source tree governed by Linus Torvalds – which is the authoritative community kernel development source code repository from which all major Linux distributions (including Red Hat Enterprise Linux) are derived. http://www.kernel.org.

7 http://rt.wiki.kernel.org

engineers have played instrumental roles in successfully incorporating upstream. All of these capabilities are included as of the mainline kernel version 2.6.18 – which forms the foundation of Red Hat Enterprise Linux 5. This table of features is not a comprehensive listing – rather it focused on the larger features.

Feature	**Mainline kernel version**	**Brief description**
BKL preemptable	2.6.8	Increase kernel locking granularity to allow more concurrency
Mutex synchronization	2.6.16	Introduces a new kernel control primitive to replace semaphores with mutexes – which have ownership properties required for implementation of priority inheritance. A lighter weight locking mechanism improving overall kernel performance.
High resolution timer infrastructure	2.6.16	Allows time to be internally repesented at nanosecond (ns) resolution.
Deterministic kernel timer event handling	2.6.16	Changed kernel timer algorithm to be O(1) deterministic on event add/remove/expiration.
Lock validator	2.6.18	Efficient runtime checking to confirm correct lock behavior. Can detect race conditions without actually hitting them. Instrumental in demonstrating correctness of realtime enhancements (as well as identifying numerous existing locking issues).
Priority inheritance futexes	2.6.18	Prevents unbounded priority inversion scenarios for user space processes – whereby lower priority processes can prevent higher priority processes from running. Requirement of RTSJ (realtime java) and telco.
Generic IRQ layer	2.6.18	Cleanup interrupt handling to factor into common code what was previously architecture specific. Laying the groundwork for later interrupt enhancements – to defer processing to threads.
Core time rewrite	2.6.18	Factor into common code prior architecture specific replication. Laying the groundwork for later dynamic

		ticks and high resolution events capabilities.
Latency tracer	2.6.18	An efficient mechanism to measure the longest latency codepaths (as they are the source of non-determinism). Useful for distinguishing kernel vs user-space latency. Allows specifying target latency thresholds and logging any that exceed. (The latency tracer is not incorporated into standard RHEL5.)

Although not explicitly itemized in the above table there are a class of capabilities included in Red Hat Enterprise Linux 5 aimed at system resource prioritization. These capabilities (also present MRG Realtime) are not new to Red Hat Enterprise Linux 5, but are listed here for completeness. These capabilities allow standard Red Hat Enterprise Linux 4 and Red Hat Enterprise Linux 5 to be tuned to provide a high level of determinism required by the vast majority of application workloads. In fact, these tuning mechanisms alone when applied to standard Red Hat Enterprise Linux releases have typically enabled us to exceed the performance and determinism of competing Linux based realtime products. We have observed that only a minority of workloads truly demand the advanced low-latency capabilities of MRG Realtime. This is why we suggest that customers first try to meet their performance objectives with standard Red Hat Enterprise Linux. If their needs for determinism are more demanding, then we suggest MRG Realtime.

- ***Interrupt binding*** – designating specific CPUs handle device interrupts.
- ***Application binding*** – restricting certain CPUs to running designated application processes.
- ***Memory pinning*** – designating that physical memory be exclusively allocated to dedicated processes.
- ***Scheduler prioritization*** – ability to designate process priorities at a fine-grained level. New to the realtime offering is the ability to set application priority above most operating system services.

Through combinations of assigning realtime application processes and critical system services to designated realtime CPUs, fine-grained system control is possible – yielding predictably low latency response times. Lower priority applications and system services are delegated to the non-realtime processors.

4.2 Beyond Red Hat Enterprise Linux 5

Having all of the enablers and realtime features itemized in the prior section included in mainline Linux was a great achievement in the delivery of Red Hat Enterprise Linux 5. This enables the general purpose

COTS offering to benefit from the associated performanceand determinism improvements. As such, many demanding workloads have their needs fully satisfied with properly tuned Red Hat Enterprise Linux 5. The Red Hat engineers continue the upstream advancement of realtime capabilities. Subsequent to the initial Red Hat Enterprise Linux 5 product launch, there have been additional key features successfully integrated upstream, and several others which continue in their development progression. The following table highlights the key realtime features that Red Hat is driving for mainline inclusion:

Feature	**Mainline kernel version**	**Brief description**
Infiniband support	2.6.20 and later	Red Hat is working collaboratively with a community of Infiniband developers from many companies. Due to the fast-paced nature of Infiniband development, the codebase in the standard Red Hat Enterprise Linux 5 product is ahead of mainstream kernel acceptance. The same Infiniband OFED 1.2 codebase enhancements from standard Red Hat Enterprise Linux 5 are also included in MRG Realtime. This infiniband support includes kernel modules as well as accompanying user space libraries.
Dynamic ticks	2.6.21	Rather than interrupt the system 1000 times per second for timing, use hardware timer interrupts to only wakeup as needed. Improves timing granularity, reduces needless frequent interrupt handling, and decreases power consumption in idle times.
High resolution events	2.6.21	Increase granularity of sleep timing from the millisecond (ms) to microsecond range (hardware dependent)– critical for time sensitive applications. Utilizes precise hardware based timer interrupts rather than imprecise software timing at the granularity of periodic interrupts. High resolution timers are utilized by the following interfaces: nanosleep, itimer, posix-timers CLOCK_MONATOMIC & CLOCK_REALTIME. (The conventional kernel HZ periodic interrupts at 1000/second continue to govern:

		poll/select and scheduler timeslices.)
CFS – completely fair scheduler	2.6.23	The objective of the CFS scheduler[8] is to provide fair interactive response times. It does away with the prior run queue, timeslice algorithm. Replacing it with a red/black tree representing runnable processes. It performs runtime calculations based on the current number of runnable processes to give each one its fair share of the compute cycles. Included in the CFS patch is a modular scheduler infrastructure. This allows for optimization of various aspects (ie, not complete scheduler replacement). One of the initial initial scheduler modules is a real-time schedule, this is first in the priority chain.
Interrupt handling in kernel threads	In progress	Converts interrupt handlers into preemptable threads. Lengthy interrupt paths are the biggest source of non-determinism. By deferring most interrupt processing to separately schedulable threads latency vastly improves as does increased prioritization control flexibility.
Converting kernel spin locks to mutexes	In progress	Spin locks are non-preemptable (often blocking interrupts) – increasing the latency of higher priority processes (reducing concurrency). This work lowers the latency of preemption points and allows the timer interrupt to occur – allowing high priority processes to run immediately.
Kernel PI (Priority Inheritance)	In progress	When a kernel thread is preempted while holding a lock, apply PI priority boost to prevent higher priority kernel threads from being stalled by lower priority threads.
Full RT-preempt	In progress	Fully preemptable kernel– all interrupt handling as threads, conversion of spin locks to sleep locks, controls to run user space processes at higher priority than kernel threads. In practice this will be

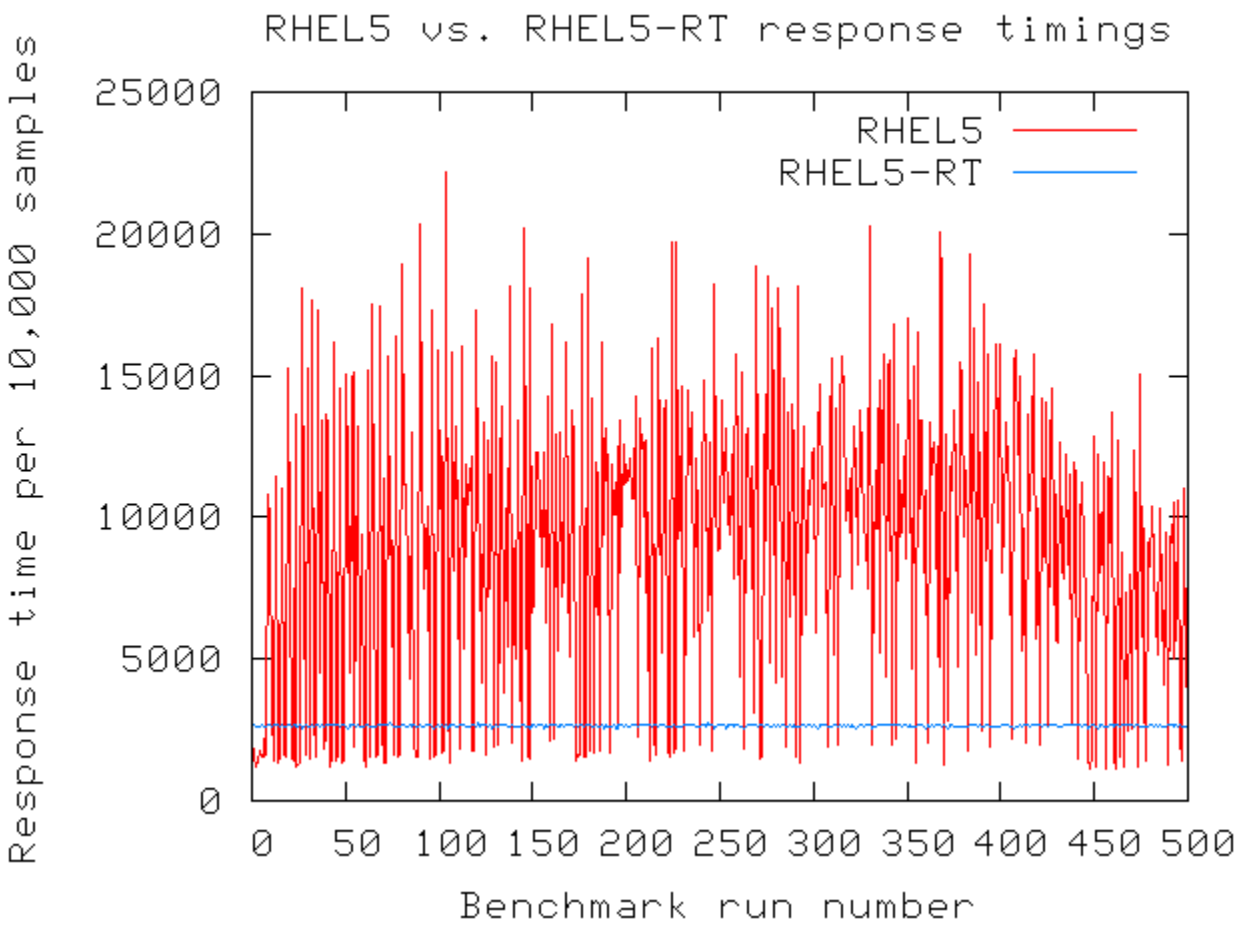
8 http://lwn.net/Articles/230574/

		decomposed to a series of changes to utilize the underlying realtime foundation put in place by all the above features. Performance optimizations are introduced for the small subset of kernel functions which are not preemptable to bound latency.

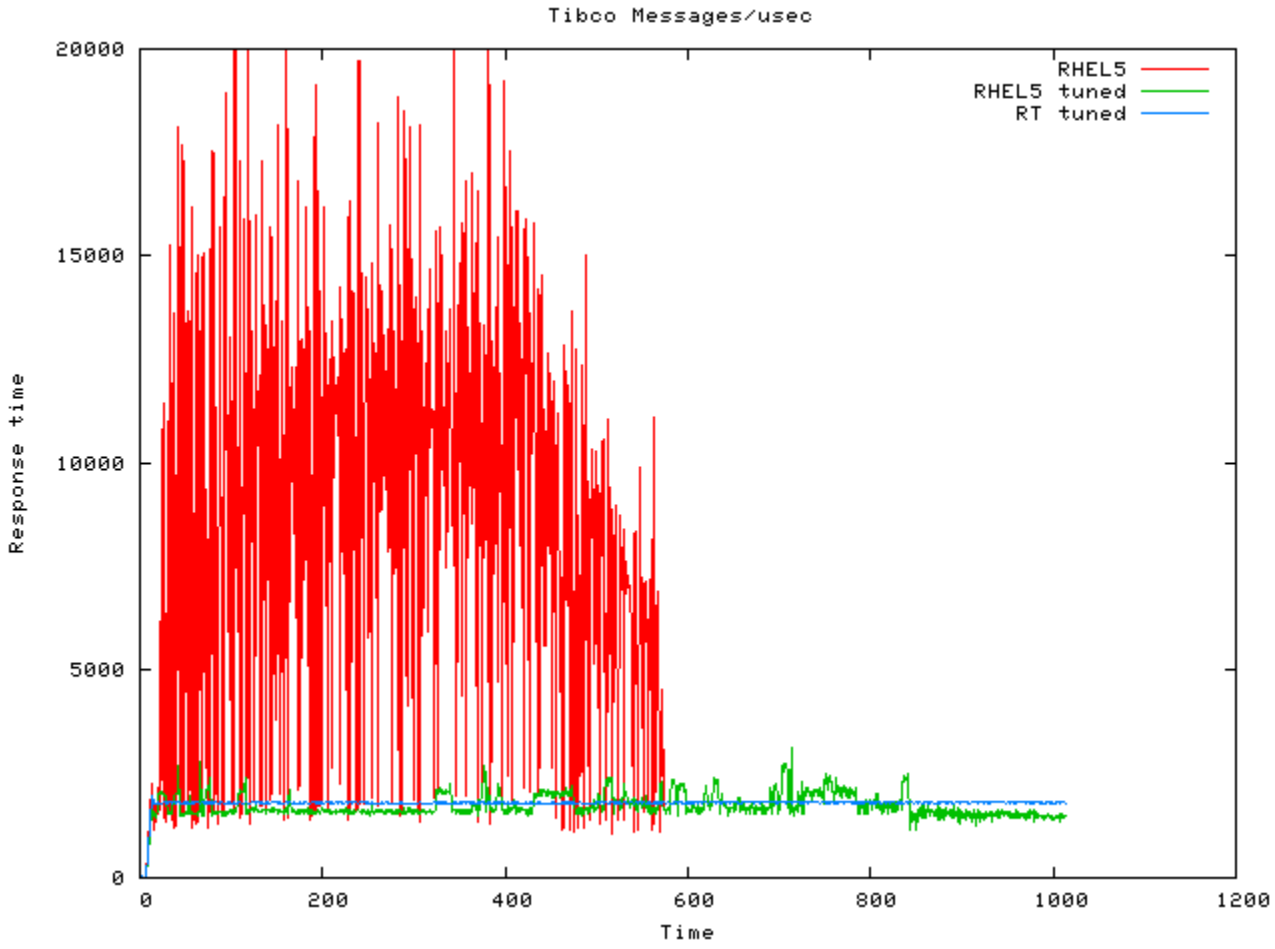
5 Performance Results

The primary objective of MRG Realtime is to introduce determinism into application workloads. The goal is to complete the user space application transactions within a finite time window – and have as few *outliers* as possible. The following graph depicts a comparison run the Tibco EMS[9] workload on a MRG Realtime kernel on a *tuned system* vs the stock Red Hat Enterprise Linux 5 kernel running on an *untuned system*. This workload is typical of applications used in the financial services industry, consisting of requests arriving over the network, the computations being run, and the results being returned. In these results you will notice the significantly more consistent transaction speed. The graph displays a 10x reduction in variability spikes. We consider this workload to be highly representative as it demonstrates an end-to-end messaging scenario.

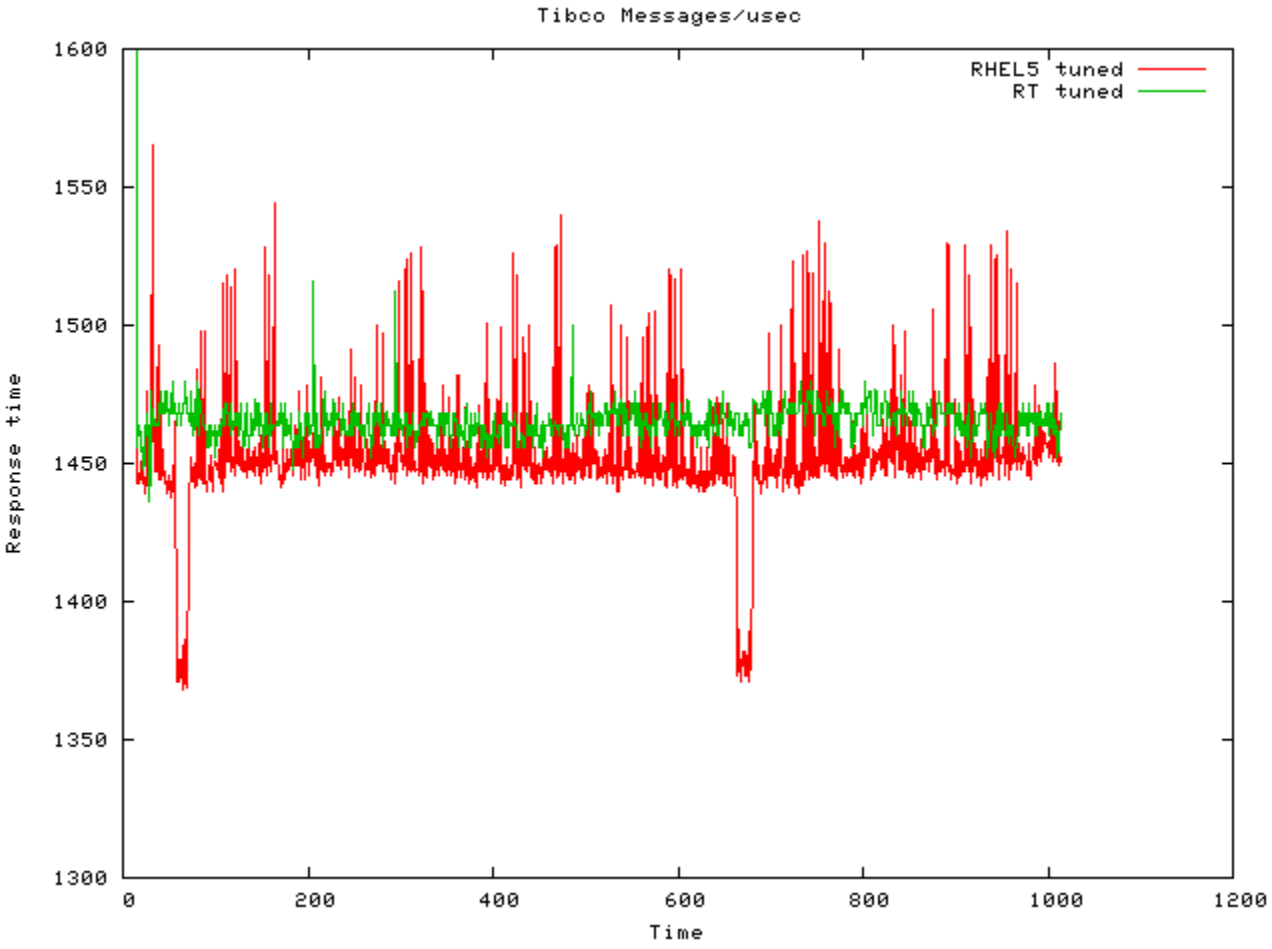
9 http://www.tibco.com/software/messaging/enterprise_messaging_service



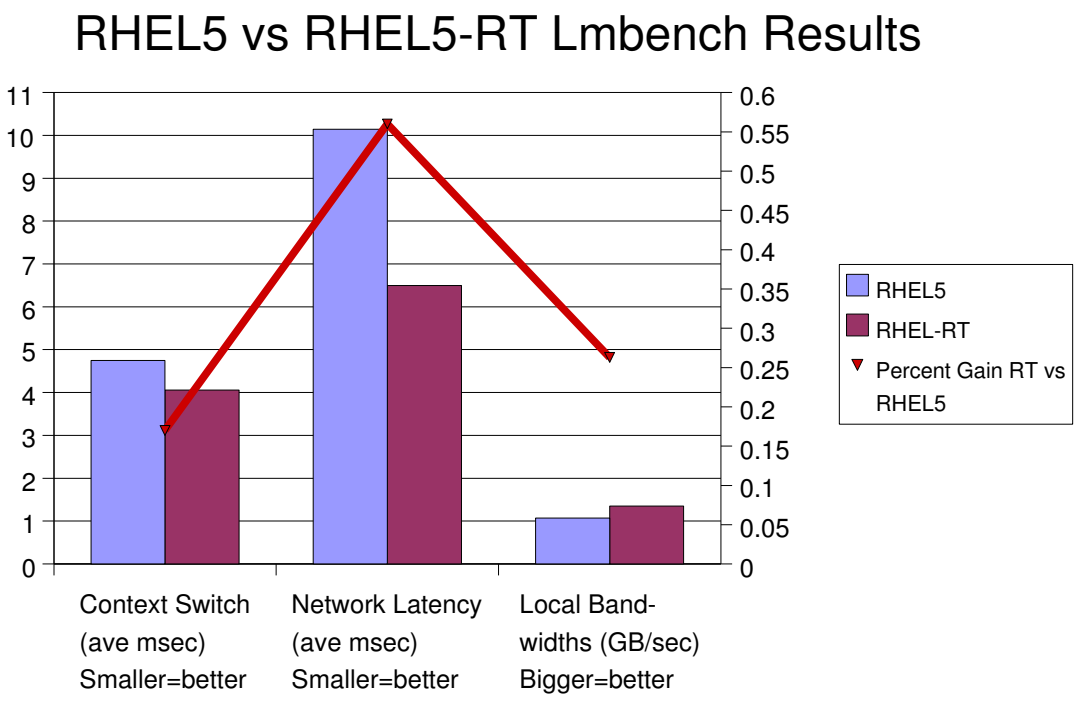
The careful reader will remark that a tuned MRG Realtime vs an untuned Red Hat Enterprise Linux 5 is an unfair comparison. Agreed. However, the reason we included the above performance chart is because other competitors' marketing collateral depicts their offerings vs an untuned standard Red Hat Enterprise Linux offering (conveniently omitting the tuning distinction). We include the above picture to illustrate that comparisons are not always apples-to-apples. Having said that, the chart below depicts results comparing an untuned Red Hat Enterprise Linux 5 vs a tuned Red Hat Enterprise Linux 5 vs a tuned MRG Realtime:



The above chart illustrates that the vast majority of determinism comes from proper system tuning. The importance of tuning is equally applicable in the case of standard Red Hat Enterprise Linux and the realtime kernel (regardless of vendor). In fact, without proper tuning, you may not see the benefits of realtime in your workload. Due to the scale of the above chart, it is difficult to see the differentiation in determinism when running the MRG Realtime kernel. To get a better view of that, the next diagram is a zoom-in comparison of a tuned Red Hat Enterprise Linux 5 vs a tuned MRG Realtime. You will notice there that there is more variability in the response times in standard Red Hat Enterprise Linux 5.



The following chart depicts performance advantages of the realtime kernel when running the lmbench micro-benchmark. Of key interest is the substantial improvement in network latency response. Both of these diagrams illustrate why MRG Realtime is particularly well suited to messaging workloads.



6 Performance Tuning & Low Latency Techniques

Be very wary of some realtime product's marketing collateral if it touts realtime kernels as a silver bullet to address any and all performance bottlenecks. Such misleading collateral commonly touts improvements in response times, but neglects to mention throughput degradation. In short, realtime is not for every workload. Red Hat's approach is to encourage customers to first perform standard system tuning practices. For the vast majority of customers, their low latency requirements are able to be achieved with the standard Red Hat Enterprise Linux distribution (ie, Red Hat Enterprise Linux 3, Red Hat Enterprise Linux 4, Red Hat Enterprise Linux 5).

A key aspect of our realtime productization includes an accumulating set of documentation in the form of HOWTO writeups. These writeups describe system tuning recommendations, such as:

- dedicating cpus to specific functions such as:
  - interrupt handling (aka interrupt shielding)
  - high priority application threads only (not kernel)
  - general kernel functions
- setting realtime scheduler priorities

- dedicating pinned memory, by pre-allocating per application, to avoid swapping
- using new latency tracing tools

Additional topics continue to be added over time. In fact, many of the low-latency tuning tips apply equally to standard Red Hat Enterprise Linux as well as MRG Realtime – well worth checking out regardless of whether you ultimately elect to use MRG.

7 MRG Realtime Productization

Red Hat is taking the same proven successful approach to delivering MRG Realtime that it has made Red Hat Enterprise Linux the industry leading Linux distribution. These steps include:

- Significant mainstream Linux community participation to drive upstream implementation.
- Select a stable product branch point from the ongoing upstream progression.
- Perform substantial testing – both internal and external beta testing involving partners and customers to shake out bugs.
- Ensure that target hardware platforms thoroughly work.
- Provide customer documentation detailing how to utilize the new capabilities – including system tuning and performance monitoring.
- Provide ongoing maintenance update releases and asynchronous errata – and ensure these fixes are fed back to the mainline community source base.
    - Bug fixes
    - Security fixes
    - Add support for new hardware
- Provide industry leading support and professional consulting services.
- Continue ongoing development in the upstream community context in preparation for advancement in later releases / updates.

MRG Realtime will be delivered as a **separate product** on top of a standard Red Hat Enterprise Linux 5 install. It will be sold and priced separately from standard Red Hat Enterprise Linux 5. The MRG Realtime software includes the following components:

- A **realtime kernel variant** – replacing the standard RedHat Enterprise Linux 5 kernel. (This is a completely new kernel development branch vs standard Red Hat Enterprise Linux 5. This new kernel is run instead of the standard Red Hat Enterprise Linux 5 kernel – it does not consist of modules which are loaded onto a standard Red Hat Enterprise Linux 5 kernel.)
- A few realtime specific utilities such as:
    - **Performance monitoring tools** – ie, Latency Tracer, and GUI used for cpu binding and interrupt pinning.
    - **Configuration utilities** – used to simplify control of kernel thread priorities based on

functionality groupings.
    - Updated versions of several utilities which are closely tied to the kernel, used for resource allocation – such as binding interrupts to certain cpus and exclusively dedicating cpus to designated user applications.
- **HOWTO documentation** – describing new capabilities, tuning recommendations and tips for performance monitoring

The user experience of deploying MRG Realtime consists of:

- First obtain entitlements and RHN register for standard Red Hat Enterprise Linux 5 and MRG Realtime
- Install standard Red Hat Enterprise Linux 5.1 (or later)
- Install the MRG Realtime packages (including the replacement kernel)
- Reboot to run the realtime kernel
- Maintenance of the standard non-kernel Red Hat Enterprise Linux 5 packages follows standard RHN delivery
- Maintenance updates for MRG Realtime will be delivered on the MRG product RHN channel.
- Support is provided via the same contacts. A single point of contact with Red Hat.

There are substantial benefits in delivering MRG Realtime as a product on top of standard Red Hat Enterprise Linux 5, including:

- **Compatibility with Red Hat Enterprise Linux 5 -** Consider that Red Hat Enterprise Linux 5 ships with over 1000 software components. In contrast the MRG Realtime product involves a small handful of about a dozen components. Hence the vast majority of Red Hat Enterprise Linux 5 is unchanged – this is largely attributable to getting the majority of the base platform enablers into stock Red Hat Enterprise Linux 5. For example, we do not replace the glibc (standard C runtime library) – hence application compatibility and standards conformance is ensured.
- **Shared maintenance and testing** – since the majority of packages are unchanged, the high-volume testing and maintenance services of Red Hat Enterprise Linux 5 remain directly applicable.
- **Broad capabilities** – Red Hat Enterprise Linux 5 includes an industry leading capability set. A premier example is the SELinux security features integrated throughout the distribution. These same SELinux security features are fully operational in MRG Realtime. This allows MRG Realtime to be unique in the Linux industry in that you don't have to make an either-or choice of security vs determinism. The MRG Realtime capabilities can not be used in conjunction with the virtualization capabilities included in Red Hat Enterprise Linux 5.[10]

10 Realtime capabilities can not be used in conjunction with virtualization due to the fact that the underlying hypervisor

8 No need for application changes

Recall that these realtime enhancements to Red Hat Enterprise Linux are contained within the kernel. This means that from an application perspective the enhancements are all *under the hood*. **No application changes are required** to benefit from the realtime capabilities.

This is not to say that MRG Realtime is a panacea. Of course, latencies introduced entirely in userspace (sub-optimal application code, unbounded java garbage collection, etc) are not eliminated from the application – nor are latencies introduced by external hardware such as network and storage. Applications which are latency bottlenecked due to kernel scheduling and interrupt handling will see benefit.

It is worth noting that there is benefit for application recompilation on Red Hat Enterprise Linux 5 (rather than using unmodified applications compiled on prior Red Hat Enterprise Linux releases). This is attributable to the Red Hat Enterprise Linux 5 glibc implementing ***pi futexes*** – which are a synchronization mechanism. These enhancements allow uncontested lock granting to be implemented entirely in user space – obviating the need for system call overhead. Since MRG Realtime uses the same glibc, these pi futex enhancements are applicable in a realtime deployment – in fact there are additional performance enhancements in the MRG Realtime kernel level for handling of contested locks.

9 Supported Hardware

As with standard Red Hat Enterprise Linux, the target hardware platforms for MRG Realtime consist of high-volume commodity hardware. In other words, there is no custom, exotic hardware required in order to utilize the realtime capabilities. There are fundamental hardware requirements in terms of latency, response times – however these requirements are readily met in most high-volume hardware shipping today.

In order to comprehensively test and validate our customer's needs, we are focusing our development and testing efforts of MRG Realtime on a subset of the overall Red Hat Enterprise Linux 5 supported hardware based on demonstrated customer need. The supported architectures include **x86 and x86-64 – both Intel and AMD** processors. We are working closely with several major manufacturers on joint testing initiatives to validate high-volume platforms demanded by realtime customers. Red Hat will be publishing the list of fully tested configurations – please provide input to your Red Hat representative to ensure your platform of interest is incorporated into our test grid.

---

does not provide true realtime determinism.

10 The broader platform picture

Today's complex IT infrastructure requires many components to work well together to form a complete application stack. The foundation layer consists of verified realtime capable hardware combined with the MRG Realtime kernel and operating system environment. There are other mid-tier components which Red Hat recognizes are essential in many deployments. Examples include:

- **Realtime Java runtime** – RTSJ compliant JVM – Red Hat is closely partnering with IBM – refer to the following section for details.
- **Messaging** – reliable, high performing, secure, interoperable messaging is a key requirement of many distributed applications. Red Hat is a primary founder and contributor to an industry consortium called Advanced Message Queuing Protocd (AMQP)[11] which is developing an open standard for messaging middleware. Red Hat is a key contributor to the corresponding open source implementation of AMQP in the Apache Qpid[12] project. This high speed messaging capability warrants mention in this realtime whitepaper as the two technologies are complimentary. The messaging feature relies on realtime features for optimal performance. These MRG Messaging capabilities will be supported on the standard Red Hat Enterprise Linux 5 platform. However they will achieve optimal deterministic performance on a MRG Realtime foundation.
- **Grid scheduling** – MRG Grid - High Throughput Computing (HTC) delivers large amounts of computing power over a sustained period of time, whereas High Performance Computing (HPC) delivers significant computing power over a short period of time. MRG Grid provides value across both HTC and HPC by offering features like: Scalability for computing the largest workloads, powerful and easy-to-use management capabilities. The ability to cycle-steal from destkop workstations and schedule to remote grids for additional computing power. The grid scheduler is able to use the more precise realtime priority scheduler to achieve predictable performance for it highest priority workloads.
- **Infiniband** – customers seeking the lowest latency messaging capabilities often utilize infiniband hardware. To meet these needs, included in MRG Realtime is kernel support and accompanying libraries and utilities for infiniband support. This includes OFED-1.2 kernel level capabilities for IB hardware drivers (mthca, ipath, ehca, amso1100, iw_cxgb3) as well as upper layer protocols (SDP, iSER, SRP, IBoIB, VNIC). Library support for IB includes: libibverbs (for direct Verbs API programming), librdmacm (to simplify connection management), OpenSM subnet fabric manager, DAT-1.2 compliant dapl libraries, and OpenMPI libraries. Our AMQP messaging

11 http://www.amqp.org/

12 http://wiki.apache.org/incubator/QpidProposal

middleware directly utilizes infiniband when present for optimal low-latency determinism.

11 Realtime Java

A large percentage of Linux realtime oriented customers have a requirement for realtime Java. In the form of an RTSJ compliant JVM – which is differentiated by enhancements for determinism in garbage collection, inter-JVM communication optimizations, and large thread count operation. Throughout the development of MRG Realtime, Red Hat has been working extremely closely with IBM to ensure that their RTSJ compliant ***IBM Websphere RealTime*** ® JVM[13] performs optimally. IBM's realtime JVM also provides ahead-of-time compilation to reduce the inherent variability of dynamic compilation. This link highlights the requirements that RTSJ has on the underlying operating system.[14]

Raytheon is an early adopter of IBM's RTSJ combined with Red Hat's kernel technology. This press release highlights the initiative.[15] As described in the press release, the United States Navy is deploying the joint realtime initiatives of Red Hat and IBM in the DDG destroyer program.

12 The Red Hat Difference – COTS Enterprise Realtime

In summary **COTS** (commercial off the shelf) **enterprise realtime** is a new way of thinking. It is much more powerful and feature rich vs traditional hard realtime – while at the same time providing better determinism vs general expectations of soft realtime.

The following are the distinguishing highlights making MRG Realtime the obvious choice for deterministic Linux workloads:

- **Mainstream approach** – Red Hat is driving the mainstream Linux community realtime implementation. Making Red Hat ideally positioned to productize and support.
- **Engineering leadership** – nobody is better positioned than Red Hat to productize and support than the team who is making it happen upstream.
- **Compatibility** – a shared foundation with stock Red Hat Enterprise Linux 5 yields unmatched application compatibility and standards conformance.
- **Broad feature set** – a shared foundation with Red Hat Enterprise Linux 5 yields unmatched breath of features (such as SELinux security) which can be used in conjunction with realtime.

13 http://www-306.ibm.com/software/webservers/realtime/ - a realtime J2SE compliant java virtual machine

14 http://www.rtsj.org/docs/OSPlatforms.html

15 http://www-03.ibm.com/press/us/en/pressrelease/21232.wss

- **Messaging** – the complimentary AMQP based open source messaging initiative will achieve maximum performance on a MRG Realtime foundation.
- **Product leadership** – Red Hat Enterprise Linux, the recognized value leader in Linux, uniquely applies the same development, testing, support and professional consulting service capabilities to the realtime product.
- **Single point of contact** – you know us from standard Red Hat Enterprise Linux deployments all these years.
- **Partnerships** – such as the close pairing of IBM's Websphere RealTime Java runtime.