1. Maximize concurrency, not just parallelism. It’s commonplace to divide a task across processor cores to take advantage of parallelism. We also look for ways to run different tasks at the same time, and try to use every bit of capacity the CPU has. Idle CPU cycles are a wasted opportunity. Our goal is to use a system’s full capacity productively to minimize latency and maximize throughput.

2. Hardware matters. We take advantage of hardware optimizations wherever possible, and aggressively build for the most advanced systems we can get our hands on. In this industry, Hardware that is cutting edge today will soon be standard issue. We also engineer for less advanced hardware: our software scales both up and down and adapts to take advantage of the optimizations available. Abstraction from the hardware is great for ease of development, and we use high-level abstraction when we can. For peak performance, though, it all comes back to the machine.

3. Memory is slow and massive multicore changes the game. We have more raw processing power than ever before, but memory speeds haven’t kept up. Memory locality matters: in our testing, we’ve seen as much as a 10x performance bump just from making sure that code runs on a processor near the memory it uses.

4. Sweat the details, again and again. It’s easy to find the big hotspots in performance. We love finding an efficient algorithm or the perfect data structure. It’s harder to find the tiny slowdowns that accumulate throughout the system. The total effect of those tiny slowdowns can be even larger than the big hotspots. There’s only one way to keep things fast, and that’s to pay attention to every line of code and every data structure.

5. Keep it simple. Fast code only matters when the code does something important. Don’t do unnecessary work or add unnecessary features. If code is too complicated to review confidently for performance or rewrite to go faster, then it’s become too complicated. If the code isn’t important enough to go under the performance microscope, then what it does isn’t important enough to be part of your system. Prefer simple solutions, and resist the urge to add convenience at the expense of performance.

6. The conventional wisdom isn’t always wise. Conventional approaches were developed for the systems available at the time, and they worked well for that environment. In today’s world, those approaches can be slow and inefficient. For example, traditional databases spend time managing indexes for faster retrieval, yet the overhead that comes with index maintenance often outweighs the performance advantages on today’s systems. Using a divide and conquer approach that avoids locks often runs faster, and we use that approach in AMPS. Conventional wisdom may say this is the wrong approach, but measurement says that the “wrong approach” is the better solution.

7. Everything works together. Applying just one or two of these principles won’t give you high performance consistency. To create high performance software, and maintain that performance as your product evolves, it’s important to apply these principles to every update and every change. It may sound simple, but the work involved isn’t trivial. There are no shortcuts, but the results are worth it. AMPS demonstrate these principles in action, and our customers reap the rewards. The ongoing changes in the computing landscape give you the opportunity to change what you’re doing and create systems that deliver more scale and performance, at a lower cost per transaction, than ever before. It’s an exciting time in high performance computing, and 60East is proud to be a part of it.

It is intended to allow the realization of the scalable high-throughput, low-latency messaging that is required in real-time deployments such as in financial services. The architecture, design and implementation of AMPS allows the exploitation of parallelism inherent in emerging multi-socket, multi-core commodity systems and the low-latency, high-bandwidth of 10Gb Ethernet.

AMPS is designed to lower the latency in real-world messaging deployments by focusing on the entire lifetime of a message from the message’s origin to its consumption by end-user clients.

AMPS 5.2 MORE POWER IN YOUR MESSAGING Now with message preprocessing, inline enrichment, enhanced aggregation and analytics — all with the ease of use and performance you've come to expect from AMPS. The secret sauce behind the most mission critical data-intensive applications in the world's largest financial institutions is now available for your applications.

Highest Performance

AMPS is designed to minimize the end-to-end processing latency in messaging deployments that serve billions of messages per day.

Unparalleled Scalability

Built with large multicore machines and low-latency 10Gb networking in mind, AMPS scales from the smallest machines to efficiently use every dollar of your hardware spend.

Built-in Analytics

With real-time computation and aggregation support, you'll be able to quickly build analytic, alerting, and event processing functionality into your own applications. AMPS lets you compute aggregates over multiple topics, including topics of different data types.

State-of-the-World Cache

An integrated current message cache allows full implementation of client side caches without the risk of inter-system drift. Fully queryable, with support for inline enrichment and transformation.

Maximum Selectivity

Filter messages by topic and/or content to prevent topic proliferation and wasted resources in delivering messages of no interest. AMPS is aware of message content, efficiently filtering directly on the message.

Integrated Security

AMPS provides a fully integrated authentication and entitlement system, including extensible, content-aware permissions. AMPS also supports SSL/TLS connections to client applications, to protect data that travels over untrusted networks.

Rock Solid Worldwide

Proven for years in global production environments where maximum stability and performance are critical. We're believers that the best route to high-availability is to deliver products free of failure. We have a serious commitment to finding the root cause of all defects and to keep delivering a better product with each release.

Transport/Message Type Agnostic

AMPS has the ability to use any message content type or network transport. Currently supported types include JSON, BSON, FIX/NVFIX, XML, BFlat, Google Protocol buffers and unparsed binary message types. Composite message types allow you to combine payloads of different types into a single payload. For networking, AMPS supports TCP/IP and websockets out of the box. An extensibility API makes it simple to add new capabilities – we’ve got more message types in the works, or get in touch with us and build your own!