Slide 1

- 60East Technologies is a company committed to building fastest real-time streaming database software and helping application developers deliver solutions that outperform their peers.
- Culminated in the creation of AMPS, which revolutionized how real-time messaging is used to build modern applications that scale into the future.
- 60East Technologies was founded in 2010 by Jeffrey and Brand.
- Jeffrey was named one of the top 10 CIOs on Wall Street by Securities Technology Monitor
- He has worked at the largest Wall Street firms in the world, where he developed infrastructures used to build high-performance, real-time electronic trading and risk systems.
- Jeffrey holds a BS in Electrical Computer Engineering from University of California Santa Barbara
- Brand is an experienced technologist that's worked for such companies as Bank of America, Morgan Stanley, and IBM
- Brings a healthy love for Big Data problems, customer satisfaction, software quality and performance everywhere he goes
- He's has a BS in Computer Science from Oregon State University

Slide 2

- So what is AMPS? Like it says in this cartoon from their website, it's a "low latency Pub/sub system with cool features" What does all this mean?
- Together Jeffrey and Brand developed a new breed of high performance low latency messaging system designed from ground up to take advantage of three big modern parallel programming techniques: multicore systems, low latency networking and solid state storage hardware.
- Advanced Message Processing System (AMPS) delivers unprecedented levels of performance and scale required for extreme low-latency and ultra scale messaging envn.
- Jeffrey describes AMPS as being a single server product that scales to handle millions of message per second by combining topic and content based publish/subscribe, a message oriented SQL database and an aggregation engine. Let's go into more detail into each of these three features.

Slide 3

- Publish–subscribe is messaging pattern where publishers of messages categorize published messages into classes rather than programming messages sent directly to specific subscribers.
- Subscribers then express interest in one or more classes and only receive messages of interest.
- This decoupling of the publishers from the subscribers allows maximum flexibility when adding new data sources or consumers allowing for better scalability
- In the publish–subscribe model, subscribers typically receive only a subset of the total messages published. There are two common forms of filtering: topic-based and content-based.
    - In topic-based system, messages published to "topics"/channels which publisher responsible for naming. Subscribers will receive all messages published to topics to which they subscribe.
    - In a content-based system, messages are only delivered to a subscriber if the attributes or content of those messages match constraints defined by the subscriber. The subscriber is responsible for classifying the messages.
- Some systems, like AMPS, support a hybrid of the two; publishers post messages to a topic while subscribers register content-based subscriptions to one or more topics.
- For example in the figure there is a Publisher sending AMPS a message pertaining to the LN_ORDERS topic. The message being sent contains information on Ticker “IBM” with a Price of 125, both of these properties are contained within the message payload itself (i.e., the message content). AMPS routes the message to Subscriber 1 because it is subscribing to all messages on the LN_ORDERS topic. Similarly, AMPS routes the message to Subscriber 2 because it is subscribed to any messages having the Ticker equal to “IBM”. Subscriber 3 is looking for a different Ticker value and is not sent the message.

Slide 4

- The State of the World (SOW) can be thought of as a database where messages published to AMPS are filtered into topics, and where the topics store the latest update to each distinct message.
- State of the World gives subscribers ability to quickly resolve any differences between their data and updated data in the SOW by querying current state of a topic, or any set of messages inside a topic.
- Topics recorded in the State of the World are also used for caching data, providing “point in time” snapshots of active data flows, providing key/value stores over data flows, and so on.
- Topics recorded in the State of the World are the underlying sources for AMPS aggregation and analytics capabilities, and the ability to store the previous state of a message is the foundation of advanced messaging features such as delta messaging and out of focus notifications.
- AMPS also provides ability to keep historical snapshots of contents of State of the World, which allows subscribers to query the contents of SOW at a particular point in time and replay changes from that point in time.
- AMPS can maintain SOW for a topic in a persistent file, which will be available across restarts of the AMPS server. SOW can also be *transient*, in which case state of SOW does not persist across server restarts.

Slide 5

- Amps support realtime aggregations
- AMPS contains a high-performance aggregation engine, which can be used to project one topic onto another, similar to the CREATE VIEW functionality found in most Database software. Aggregation engine can join input from multiple topics, of the same or different message types, and can produce output in different message types. Allows for projections/groupings of data
- In latest addition of AMPS, Amps 5.2, an additional feature has been added called aggregated subscriptions. Gives subsribers new capability to create custom aggregations and projections on AMPS topics – with no configuration necessary, great for when creating view not practical.
- These are essentially private views for an individual subscription. You no longer have to reconfigure and restart AMPS to test a different calculation, or add a full view for a subscriber that needs different data – but only for a few days at the close of the month. When a subscriber has unique needs, aggregated subscriptions can give that subscriber a unique view.
- Example where I found this useful was dynamically creating graphs on the fly. I used amps to mimic complex queries created from selecting different fields associated with the graph- the axes of the graph, groupings and filterings. It’s not practical to create a view for every possible graph because too many different combinations. Aggregated subscriptions allow a developer to dynamically decide on the view to be created.

<u>Slide 6</u>

- Who used AMPS? Other than me in my summer internship, in use by major Fortune 500 companies
- AMPS drives the most data-intensive mission-critical applications such as trading desks, analytic grids, reporting applications, view servers, workload management, and more
- What these applications have in common is a need for reliable high-volume, low-latency messaging with advanced features that go beyond simple pub-sub or point-to-point messaging.
- AMPS makes it easy to build applications that provide the lowest levels of latency and highest overall throughput by providing built-in features such as historical replay, a real-time database with continuous queries and aggregation, focus tracking, incremental message updates and other advanced features that typically require extensive custom processing in your application.

Slide 7

- 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.
- Hardware matters. We take advantage of hardware optimizations wherever possible, and aggressively build for the most advanced systems we can get our hands on.
- 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:
- Sweat the details, again and again. It’s easy to find the big hotspots in performance. It’s harder to find the tiny slowdowns that accumulate throughout the system.
- Keep it simple. Fast code only matters when the code does something important. Don’t do unnecessary work or add unnecessary features.
- 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.