A REAL-TIME CLOUD FOR THE INTERNET OF THINGS

Ken Birman
Cornell University
THE INTERNET OF THINGS: A BIG DEAL

Smart power grid...
Smart highways to guide smart cars...
Smart homes

Huge market

Scale demands a cloud with special properties
A “SMART” POWER GRID?

Today’s grid uses mostly 1980’s technology

Human-controlled... robust, but struggling with renewable energy.

Core idea: Machine learning could bring transformative flexibility
FORMS OF MICROGENERATION

Solar panel on the rooftop

Battery in the wall

Heater or A/C unit that we can activate early or delay

➢ Think of “delayed power use” as a new form of power: “schedulable” demand!
SMART METERS

Like attaching an iPhone to the wall in the utility closet

Knows what’s going on, regulates power consumption for heavy power-use devices like A/C, heater, etc.

Ideally, the “endpoint partner” for the more centralized optimization codes that work to balance supply and demand
The intelligence lives in the cloud. The external infrastructure is mostly dumb sensors and actuators.
GRIDCLOUD (AND SIMILAR SYSTEMS) MOTIVATE MY RESEARCH AGENDA

Build on today’s cloud, but bring super-fast consistency solutions to bear on these needs

Outcome: a nimble real-time cloud with consistency

Best of all: not only does our stuff work better, it is way faster too
FORMS OF CONSISTENCY

One issue is real-time consistency
- After an event occurs, it should be rapidly reported in the cloud
- And anyone sharing the platform should soon see it

Another issue is replication consistency
- Replicate for fault-tolerance and scale
- But all replicas should have identical contents (within a brief delay)
FROZEN FRAME FS

Real-time file system for secure, strongly consistent data capture
- Normal file system but understands real-time timestamps
- Offers optimal temporal precision plus Chandy-Lamport consistency

Incredibly high speed
- Leverages RDMA for network line-speed data transfers
- NVRAM (SSD or RAID disks) for storage persistency
We simulated a wave and sampled it. Like taking photos of individual grid cells (squares). Then streamed the data to our cloud-hosted historian, one stream per sensor. Then reconstructed the wave from the files and created these gif animations.
CONSISTENT TIME MATTERS!

Real-time applications need cloud services that understand time
- Deep-learning on temporally fuzzy data will give incorrect output
- Operators and systems using stale data will make poor decisions

With Freeze Frame FS, we can start to run powerful file-oriented analytics that understand data, and that keep up in real-time
HOW DOES FFFS WORK?

FFFS CAPTURES DATA PLUS TIME

INTERNALLY, KEEPS A SCALABLE, FAULT-TOLERANT MEMORY-MAPPED DATA SET INDEXED BY TIME
READING AND WRITING TEMPORAL DATA

- Read “now” (standard POSIX API and behavior). FFFS finds and returns the data applicable at the time you requested:
  - POSIX applications read from a directory named by desired time
  - … or can use a non-POSIX API with time as an extra parameter.

- Writes: time extracted from data records (plug-in tells us how)
  - … or we can use “platform time” when the write occurred
  - … or you can just tell us using a non-POSIX API
RDMA RELIABLE UNICAST

RDMA: Direct **zero copy** from source memory to destination memory

On 40 Gb/s network, RDMA reaches 36Gb/s. TCP/IP hits 250Mb/s…

- This is a comparison with TCP on IP.
- Mellanox and Intel offer versions of TCP that use RDMA if available.

Like TCP, RDMA is reliable: if something goes wrong, the sender or receiver gets an exception. This only happens if one end crashes.
When RDMA hardware is available, this lets us blast past any file system you’ve ever seen!

- Our experiments run on Mellanox 100Gb/s routers
- We’ve worked with both ROCE (Ethernet) and Infiniband and got fantastic performance in both cases

When RDMA hardware isn’t available we just use TCP (SoftROCE) but even in this mode, run faster than standard cloud file systems
MORE CORNELL RDMA STUFF…

**RDMC:** Reliable data replication at insane speeds

**SST:** A new simple form of shared memory for rack-scale systems

**Derecho:** \{RDMC+SST\} used to create ultra-reliable replicated data with virtual synchrony groups, multicast, Paxos

… and it all works, open source.
RDMC: AN RDMA MULTICAST

Binomial Tree

Binomial Pipeline

Final Step
HOW FAST IS RDMC?

Up to 16 copies we peg the optical network...

Then slow degradation due to link contention (idea: topologically aware solution could avoid this)

RDMC should eventually scale to thousands of replicas

<table>
<thead>
<tr>
<th>Bandwidth (Gbps)</th>
<th>Group Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>6</td>
<td>128</td>
</tr>
<tr>
<td>4</td>
<td>256</td>
</tr>
</tbody>
</table>

Using Mellanox 20Gbps RDMA on Infiniband

<table>
<thead>
<tr>
<th>Bandwidth (Gbps)</th>
<th>Group Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>20Gb/s = 2.5GB/s</td>
<td>4</td>
</tr>
<tr>
<td>100Gb/s = 12.5GB/s</td>
<td>256</td>
</tr>
</tbody>
</table>

20Gb/s = 2.5GB/s... 100Gb/s = 12.5GB/s
**HOW FAST IS RDMC?**

Up to 16 copies we peg the optical network...

Then slow degradation due to link contention (idea: topologically aware solution could avoid this)

RDMC should eventually scale to thousands of replicas

---

Using Mellanox 100Gbps RDMA on ROCE (fast Ethernet)

20Gb/s = 2.5GB/s… 100Gb/s = 12.5GB/s
**HOW FAST IS RDMC?**

Up to 16 copies we peg the optical network...

Then slow degradation due to link contention (idea: topologically aware solution could avoid this)

RDMC should eventually scale to thousands of replicas

*Using Mellanox 100Gbps RDMA on ROCE (fast Ethernet)*

20Gb/s = 2.5GB/s.... 100Gb/s = 12.5GB/s

Memcpy runs at 30Gbps = 3.75GB/s
A SECOND EXAMPLE: SST

A simple shared table implemented with RDMA

Key idea: each element of a rack-scale system owns one row in the table. The row format is defined by a C++ 11 struct that has well known sizes, “plain old data”

If you update “your” row, the data is pushed to other nodes

Then we support events: “on (predicate) do { function }”

- Current version: close to 1 million events per second with 8 nodes
- Runs on one-sided RDMA writes (or reads, but writes are faster)
- Same memory model as for “volatile” variables in C++
Derecho is a group communication infrastructure.

Programs can attach to groups ("join") them.

Data is replicated by doing a group send operation.
A PROCESS JOINS A GROUP

At first, P is just a normal program, with purely local private variables.

P still has its own private variables, but now it is able to keep them aligned with track the versions at Q, R and S.

... Automatically transfers state ("sync" of S to P,Q,R)
   Now S will receive new updates
A PROCESS AS A GROUP MEMBER

All members see the same “view” of the group, and see the updates (multicasts) in the identical order.
All members see the same “view” of the group, and see the updates (multicasts) in the identical order.
**CONSISTENCY MODEL: VIRTUAL SYNCHRONY + PAXOS**

- Virtually synchronous run is indistinguishable from behavior of a non-replicated object that saw the same updates (state machine replication)
- Persistent virtually synchronous runs are the same as Paxos
DERECHO = RDMC + SST

Derecho group with members \{A, B, C\} in which C is receive-only

DNS: myGroup.somewhere.org | 123.45.67.123:6543 (Leader: A)

\[ V_3 = \{ A, B, C \} \]

DNS record and current view, showing senders

<table>
<thead>
<tr>
<th></th>
<th>Suspected</th>
<th>Proposal</th>
<th>nCommit</th>
<th>Acked</th>
<th>nReceived</th>
<th>Wedged</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>4: -B</td>
<td>3 4 5</td>
<td>3 0 T</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
<td>3 3 4</td>
<td>4 0 F</td>
</tr>
<tr>
<td>C</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
<td>3 3 5</td>
<td>4 0 F</td>
</tr>
</tbody>
</table>
... INSANELY FAST REPLICAITION

As a multicast: making 256 replicas is only 3x slower than one!

10,000x faster than older multicast libraries like Ken’s Vsync.

To SSD: Scale and performance that blows away standard Paxos and doubles what Microsoft’s record-setting Corfu system reports
DERECHO IS A MULTICAST AND PAXOS

As a multicast, moves data from DRAM to DRAM at insane speeds
  ➢ Reliable, totally ordered, can use the “view” to coordinate

As a storage solution, moves data to NVRAM (SSD disk)
  ➢ Now it has exactly the properties of Lamport’s Paxos protocol

Derecho.codeplex.com
HOW FAST IS DERECHO?

Using Mellanox 100Gbps RDMA on ROCE (fast Ethernet)
20Gb/s = 2.5GB/s…. 100Gb/s = 12.5GB/s

RDMC

Derecho as a multicast protocol
Guarantees are exactly the same as for Paxos (Corfu)

Basically, a replicated persistent log on SSD storage units

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Speed of SSD (MB/s)</th>
<th>Derecho (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube video (mp4)</td>
<td>67.7645</td>
<td>63.2405</td>
</tr>
<tr>
<td>Random bits</td>
<td>72.1108</td>
<td>67.6758</td>
</tr>
<tr>
<td>Facebook web page</td>
<td>126.451</td>
<td>112.909</td>
</tr>
<tr>
<td>Repeating text data</td>
<td>457.55</td>
<td>300.398</td>
</tr>
<tr>
<td>All Zeros</td>
<td>471.104</td>
<td>292.986</td>
</tr>
</tbody>
</table>

4-member group, 4MB objects
SO... WHAT ABOUT CONTAINERS?

Container trend is the key to the game!

Virtualization is at odds with RDMA: very hard to securely share RDMA NICs in a virtualized setting with enterprise VLANs etc.

Containers strip away the virtualization layer and enable us to leverage these kinds of insanely fast data sharing, replication and retrieval technologies.
The Internet of Things revolution is extremely promising for all of us.

Huge amounts of data will need to be captured, persisted. Many obvious roles for non-volatile memory.

Containers will win because of their lower overheads and ability to leverage RDMA and NVRAM hardware more effectively.
The Internet of Things is coming... and will bring extreme scale

We need to learn to move big data in real-time with strong consistency properties

Derecho and Freeze Frame FS are proof that if we leverage the hardware, especially RDMA, it can be done