Building on Lessons Learned from over a Decade of MRNet Research and Development

(SC2003 ⇨ SC2013)

Barton P. Miller

University of Wisconsin

bart@cs.wisc.edu

Dorian C. Arnold (UNM), Michael J. Brim (ORNL),
Philip C. Roth (ORNL), Evan Samanas (UW),
Benjamin Welton (UW), Bill Williams (UW)

Paradyn

THE UNIVERSITY of WISCONSIN MADISON

"I think that I shall never see
An algorithm lovely as a tree."

Trees by Joyce Kilmer (1919)

25 And Moses chose able men out of all Israel, and made them heads over the people, rulers of thousands, rulers of hundreds, rulers of fifties, and rulers of tens.

כה וַיִּבְחַר מֹשֶׁה אַנְשֵׁי-חַיִל מִכָּל-יִשְׂרָאֵל,
וַיִּתֵּן אֹתָם רָאשִׁים עַל-הָעָם--שָׂרֵי אֲלָפִים
שָׂרֵי מֵאוֹת, שָׂרֵי חֲמִשִּׁים וְשָׂרֵי עֲשָׂרֹת.

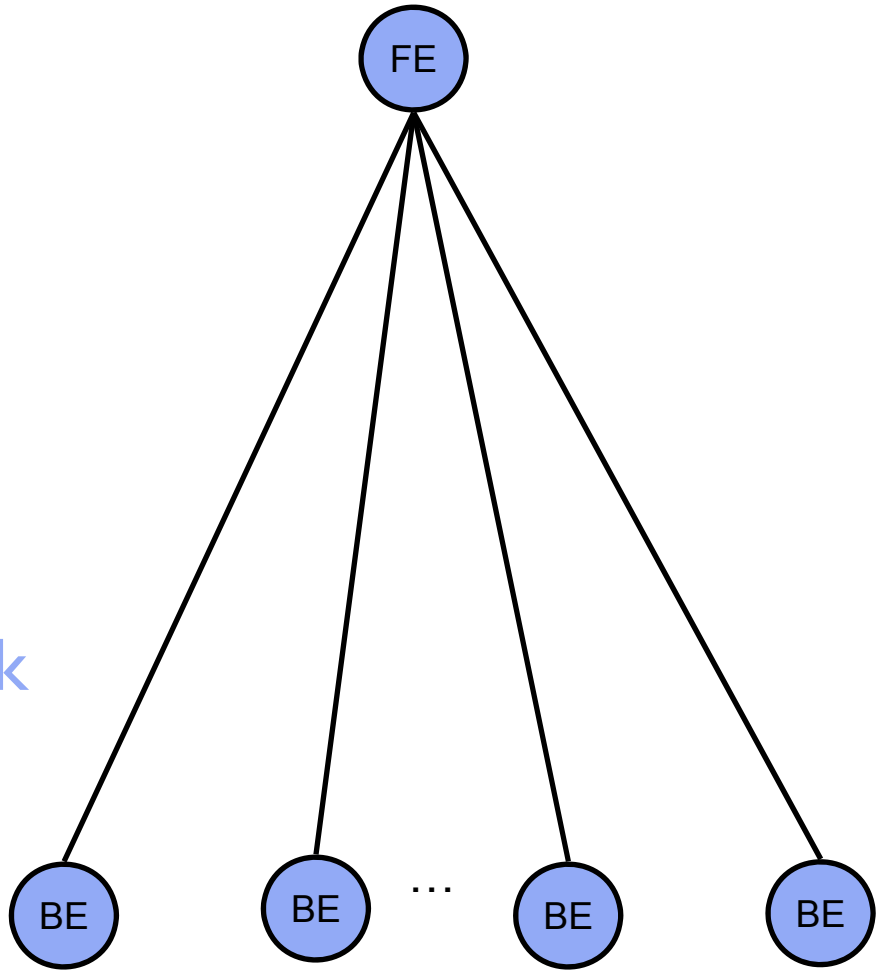
Exodus 18 שְׁמוֹת

If you can formulate the problem so that it is hierarchically decomposed, you can probably make it run fast.

Common System and Tool Scenario

- Key tasks:
  - Application Control
  - Data collection
  - Data centralization/analysis

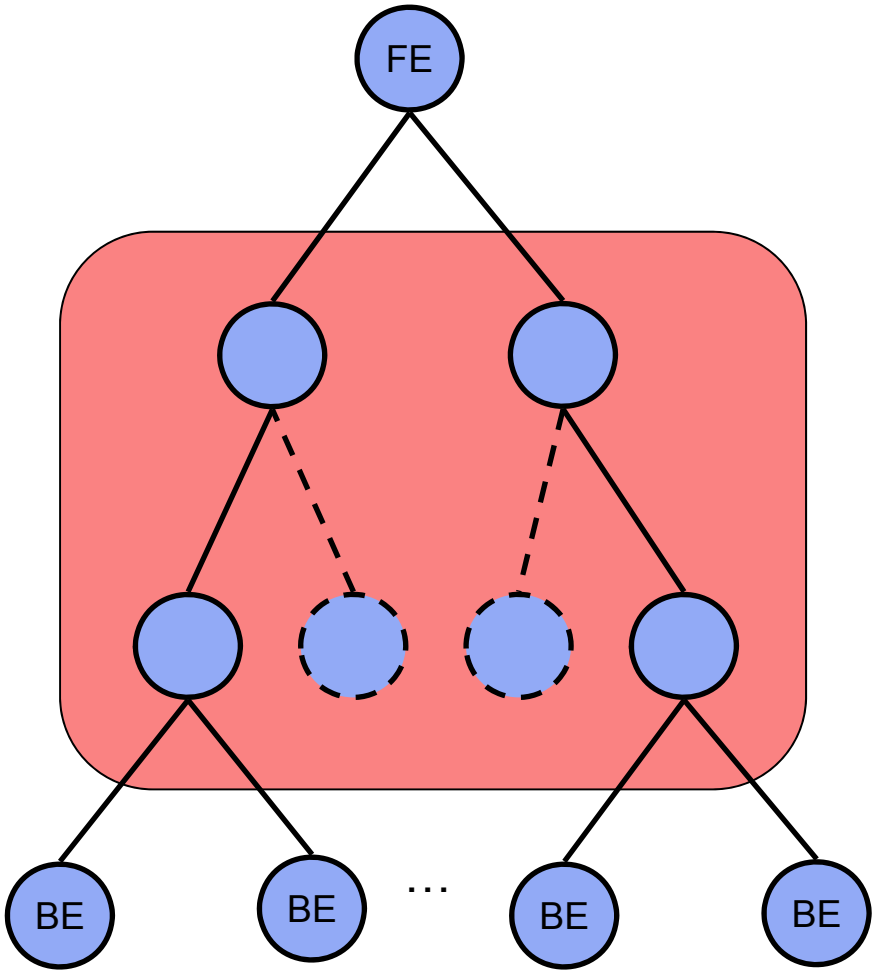
- As scale increases, front-end becomes bottleneck



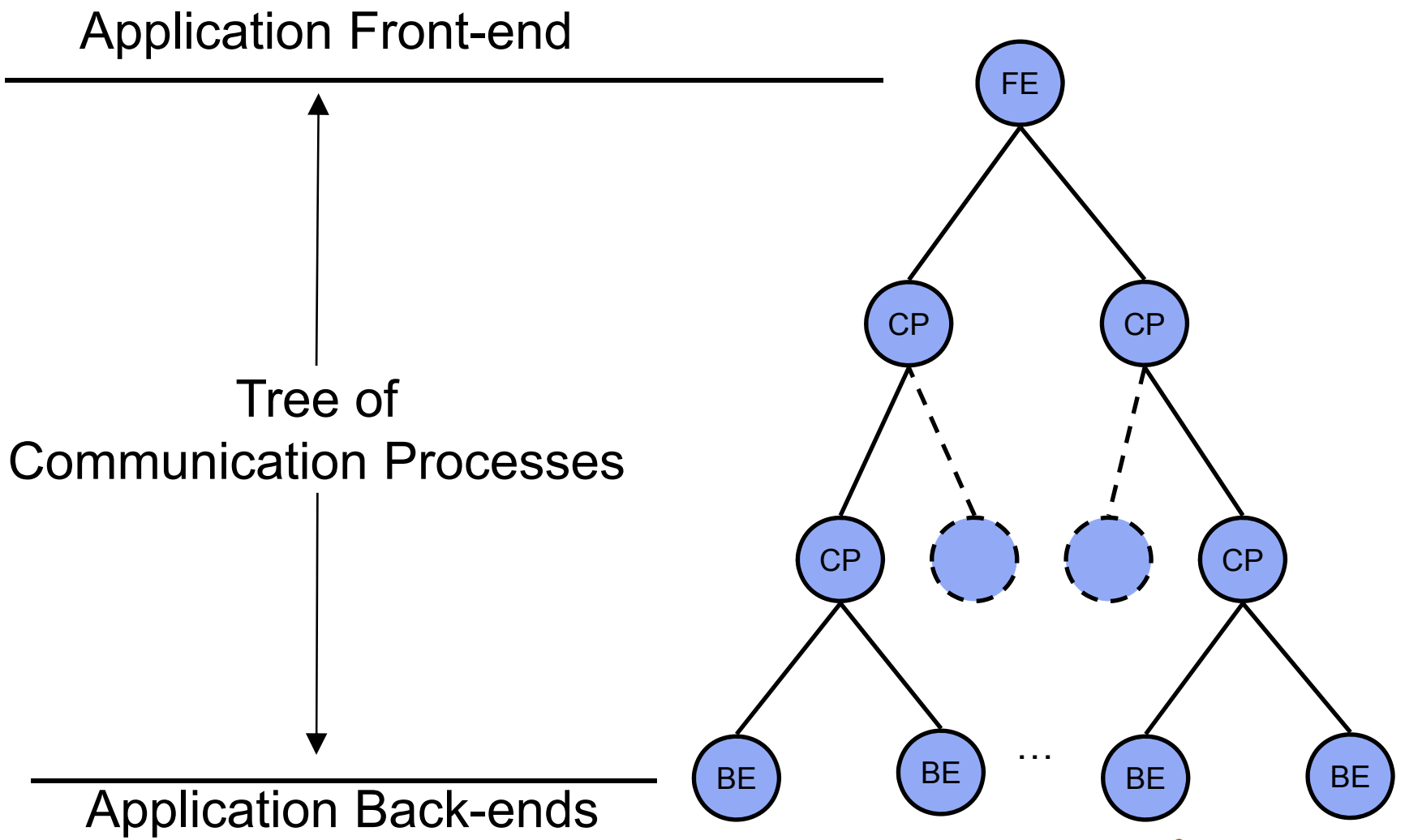
Tree-Based Overlay Network (TBŌN) Model

TBŌNs provide:

- Scalable multicast
- Scalable gather
- Scalable data aggregation
- Natural redundancy



TBŌN Model


Application Front-end
Tree of
Communication Processes
Application Back-ends
FE
CP
CP
CP
CP
BE
BE
…
BE
BE

Problem Statement

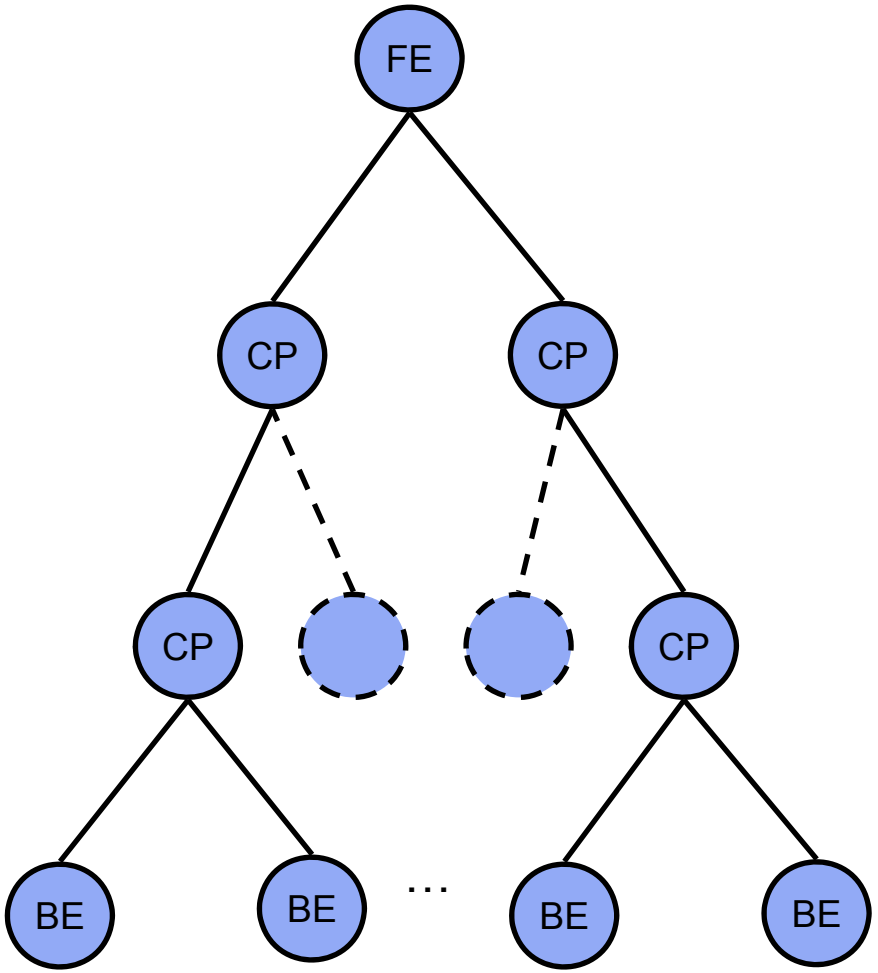
Whatever we build should be (and is):

- **Scalable** – handle ~~100,000's~~ of nodes **1,000,000's**
- Multi-platform - Cray, IBM BG, Linux clusters, and even Windows
- **Reliable** – Automatic fault recovery
- **Flexible** – Target a wide variety of systems, tools, applications, and architectures
- **Customizable** – Easily extend to new algorithms and requirements

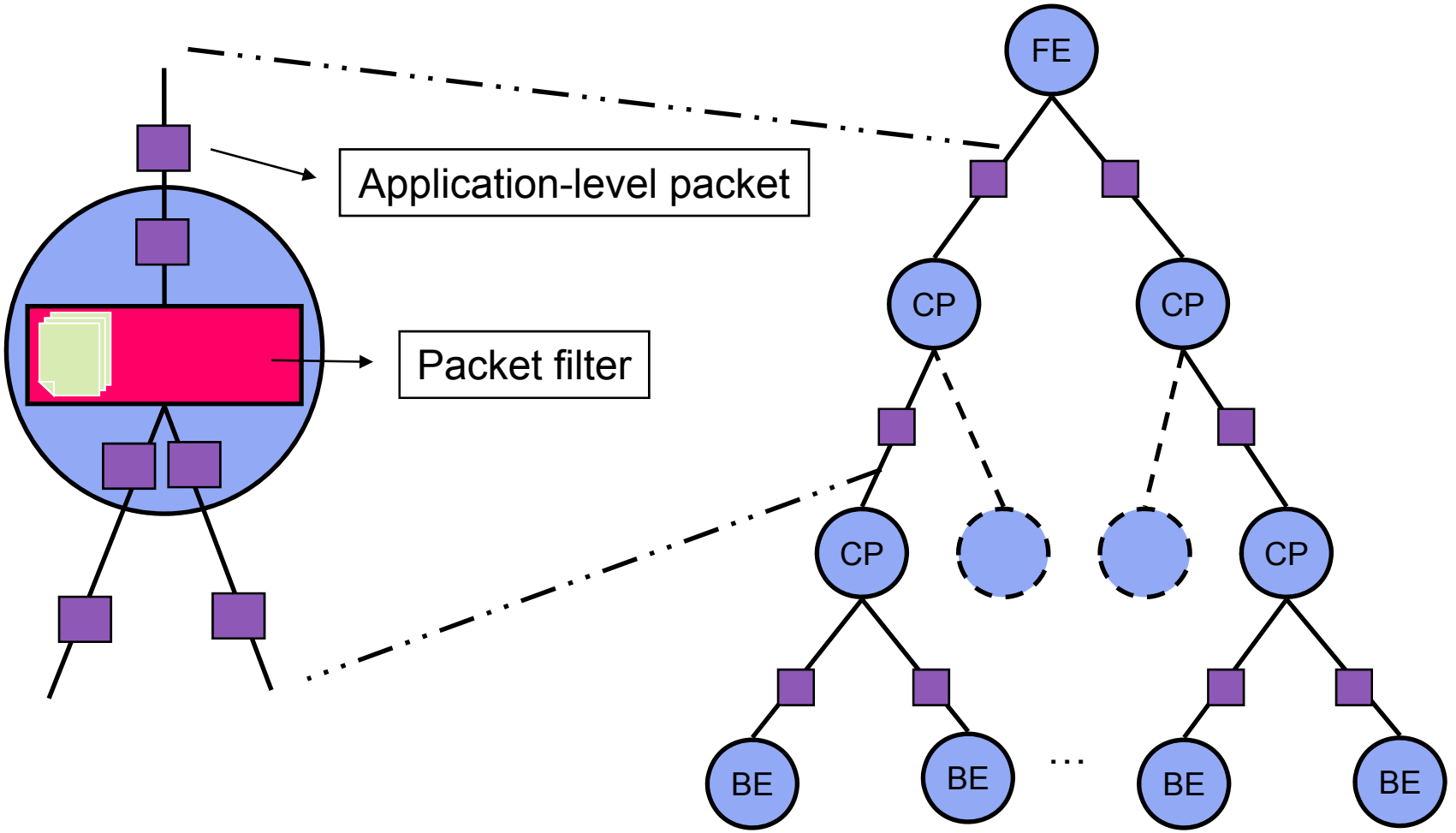
MRNet: An Easy-to-use TBŌN

Efficiency:

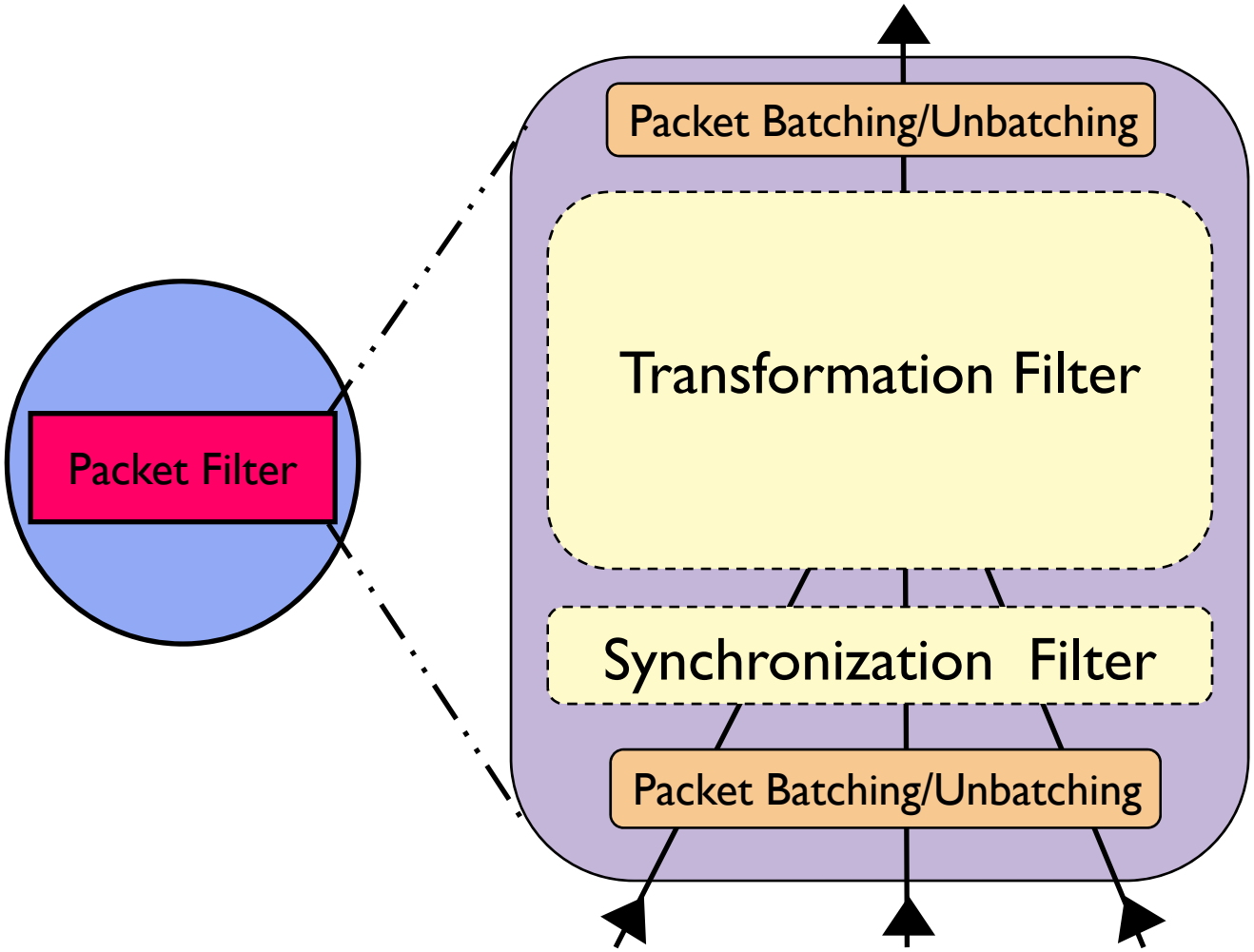
- Zero-copy paths
- Scatter-gather
- Binary data representation.



MRNet: An Easy-to-use TBŌN



MRNet Filters



Lesson: Linear scaling is evil

- O(n) is evil: at scale, any linear time operation will destroy your performance …
- … even if it's just bits.
- If there is any linear algorithm or data structure in your code, it will become painfully obvious as you scale up.

Lesson: Balance is crucial

- We all understand that the cost of the slowest component can determine the speed of all the others.
- For a tree, that means an ideal algorithm must have:
  - Similar compute time for each node at a given level of the tree;
  - Similar compute time for each level in the tree;
  - Constant data size as results propagate up the tree.

Lesson: Compute where it scales: at the leaves

- The most computing power is at the leaves. If possible, process your data there: the number of backends scale with the number of backends (duh).
- Leave the data at the leaves. If you don't need all the data at once, fetch it on demand (lazy evaluation)
- This means to carefully distinguish between group operations, which aggregate nicely, and individual operations.

Lesson: Scalable computation needs scalable viz

- Computing over very large problems efficiently is only the start …
- … you need to be able to scalably view your results.
- This might mean good summarization, clever graphics, or ways of identifying equivalence classes and selecting representatives.
- STAT was an elegant case where the computation data structure and visualization were naturally the same.

Surprise: 1K limit on `select` file descriptors

- We had a notification path directly from backends to the frontend. And found that you get a silent error if your **`fd_set`** is bigger than 1K.
- At the very bottom of the man page you see:

  "An fd_set is a fixed size buffer. Executing FD_CLR() or FD_SET() with **a value of fd** that is negative or is equal to or **larger than FD_SETSIZE will result in undefined behavior**."
- Shouldn't be doing this in the first place, but will work with the **`poll`** system call instead.

Surprise: Your scaling can overload others….

- MRNet can load filter functions from the file server as a shared object …
- … which can cause a file access storm.
- Solution is to distribute such items through MRNet itself …
- … i.e., use your own infrastructure to solve your scalability problems.
- Note that just checking a file descriptor to see if it's local or on the server can cause the storm (addressed by LLNL FGFS)

Surprise: Keeping topology up to date is tricky

- MRNet reports topology changes up the tree, then multicasts them downward.
- When back-ends start separately, like on the Cray, each attach caused a topology update …
- … causing a storm of updates up and down the tree.
- Solved by the creation of a separate topology update stream that gathers the connect updates and distributes them once.

Surprise: Termination is hard

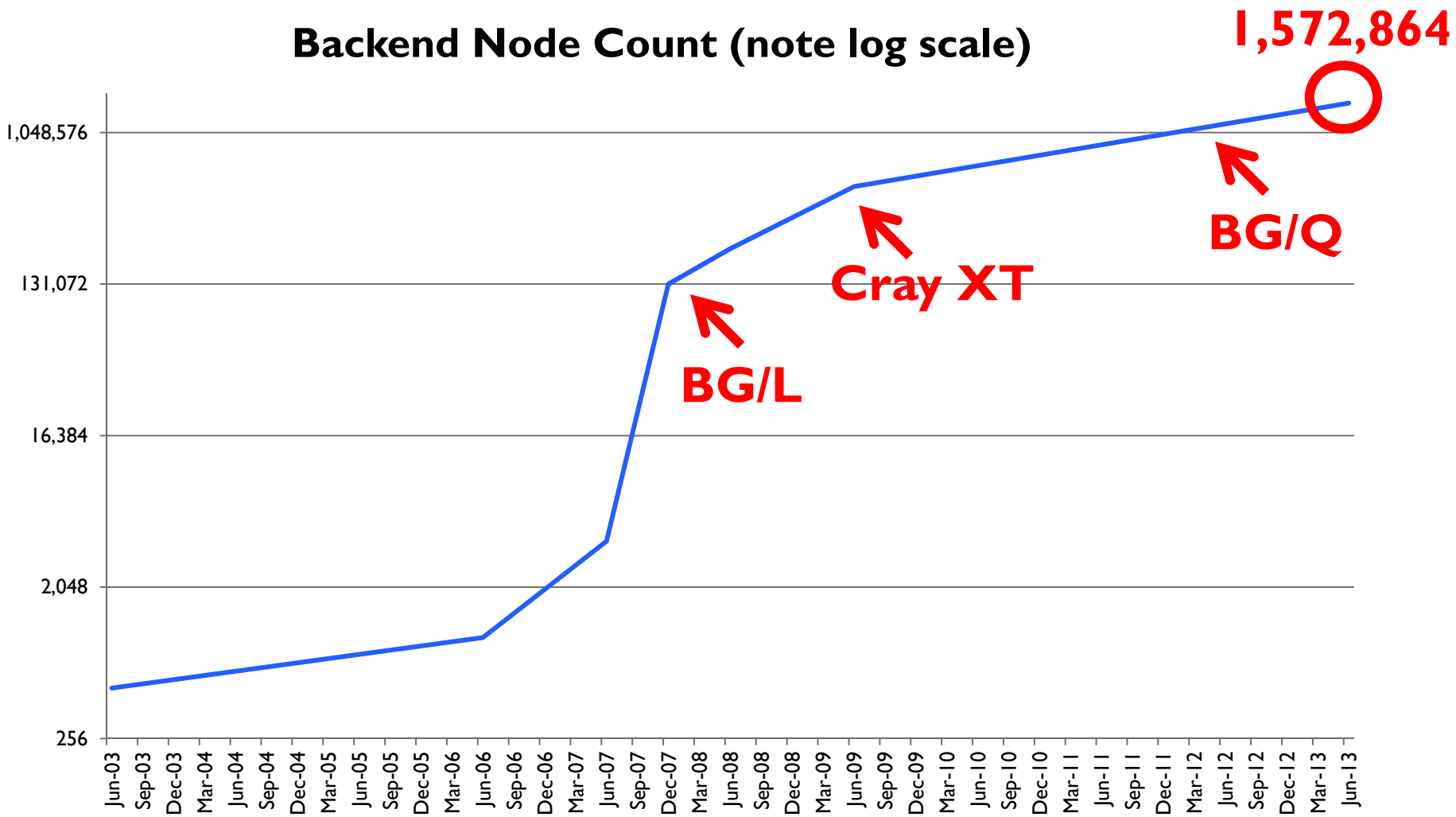
This shouldn't really be a surprise, and the distributed systems theory folks have told us this again and again.

- Races: On Cray, when an application process terminates, aprun starts killing off our MRNet CP threads, one at a time. Other threads get confused and leave lots of core dumps around. The solution is careful design and checking.
- With the fault tolerant features turned on, the tree kept repairing itself while it was trying to shutdown! Solution is knowing when you are in shutdown mode.

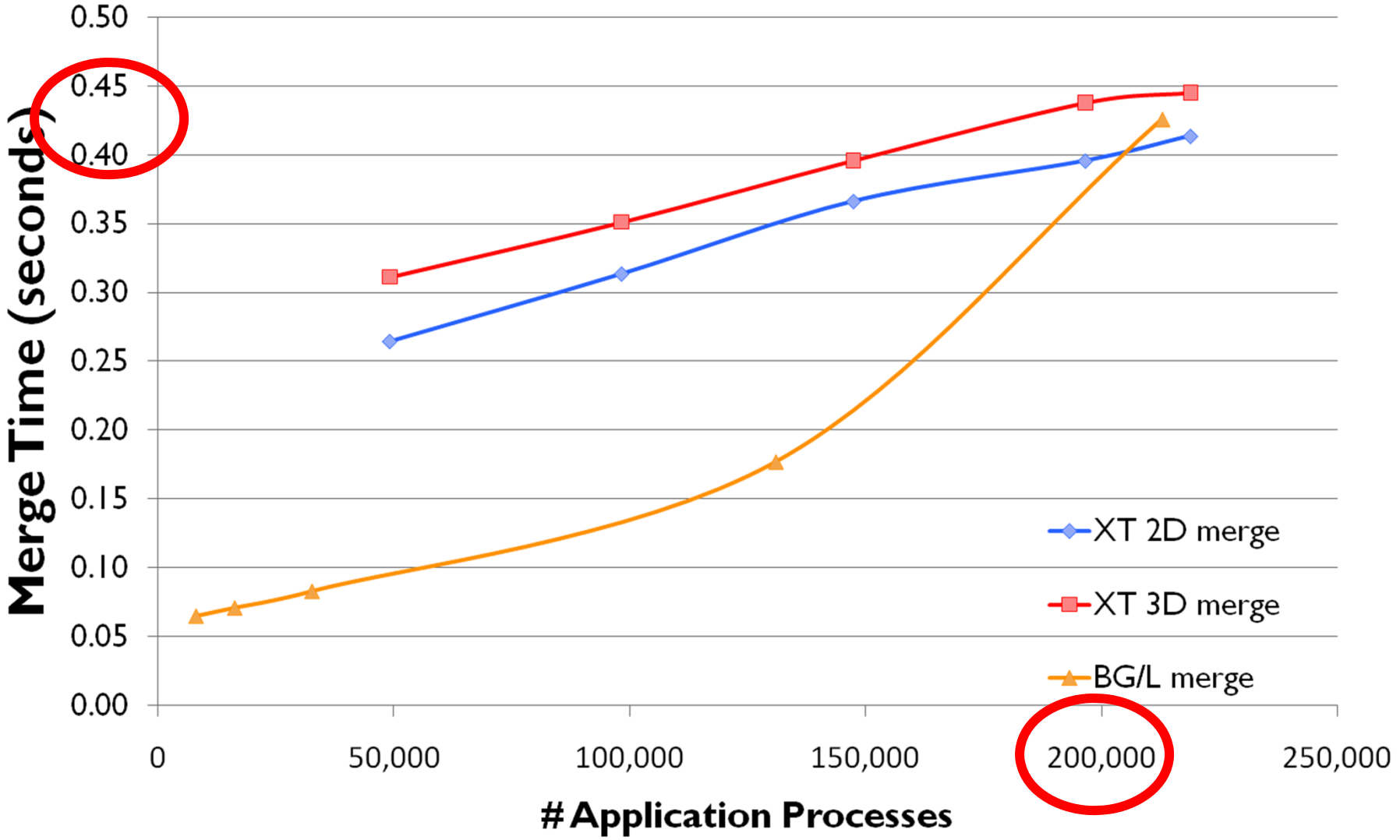
Surprise: The built-in services don't always scale

- Things like job start-up should be efficient and scalable on leadership class systems. Sadly, you find suboptimal, even linear performance.
- Solution is to do it ourselves, using our own tree whenever possible.

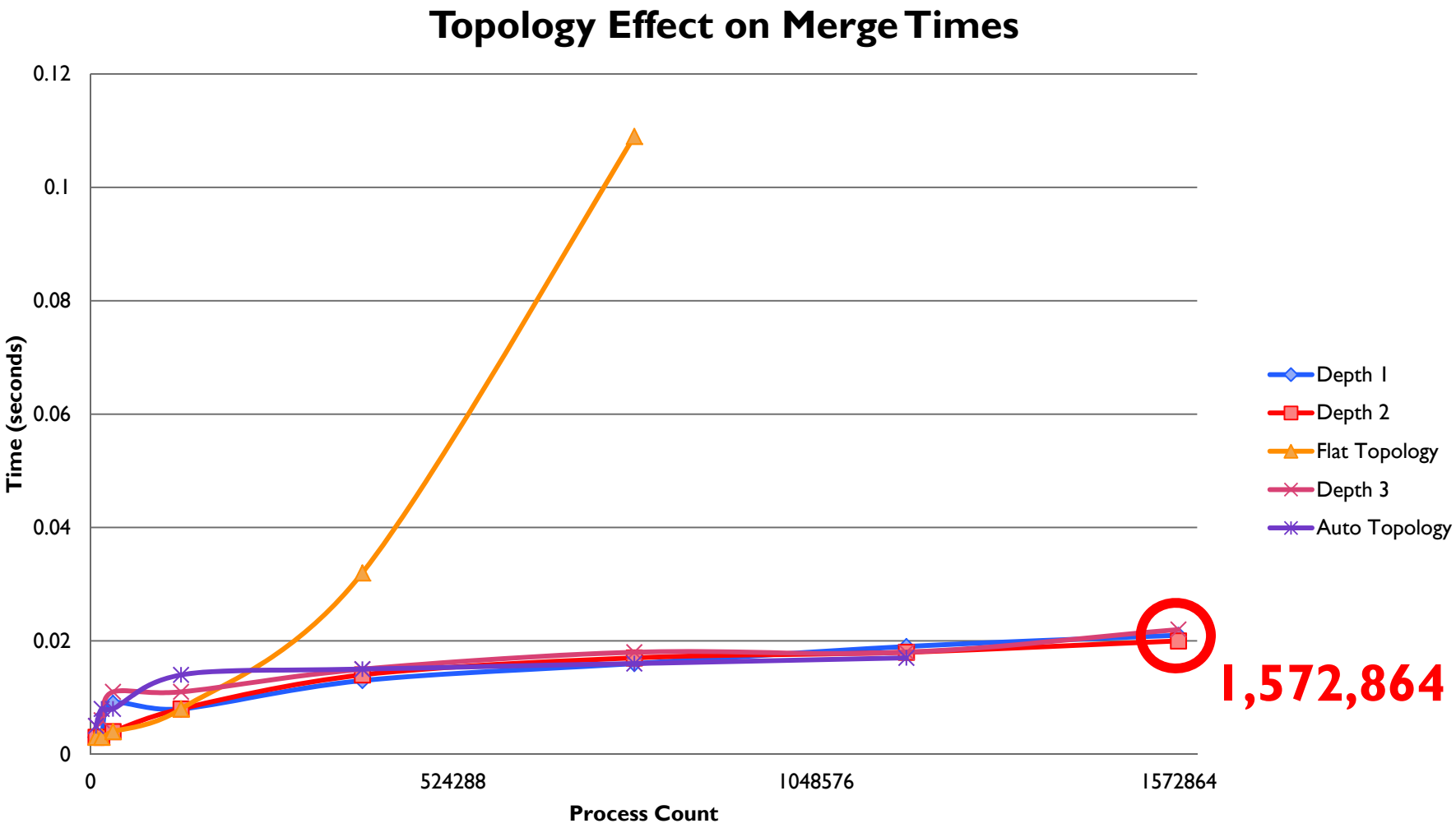
A History of MRNet Scaling Results



STAT Merge Performance



STAT Merge Performance on LLNL's IBM BG/Q



MRNet in the Wild

MRNet is becoming common infrastructure. Some deployed uses:

- Paradyn - Wisconsin, Maryland (the original MRNet target)
- STAT - LLNL, Wisconsin
- ATP - Abnormal termination Processing, Cray
- Totalview - RogueWave
- pcp, pgrep, psync, ptail, ptop - Wisconsin (TBON-FS)
- Ganglia - Wisconsin (TBON-FS)
- LIBI - New Mexico, LLNL
- FGFS (Fast Global File Status) - LLNL, Wisconsin
- Component-Based Tool Framework (CBTF) - Krell
- Tau over MRNet (ToM) - Oregon, Wisconsin
- CEPBA Tools - Barcelona Supercomputing Center
- Comparative Debugger - Monash, Cray
- System Monitoring - ORNL

Contact us:

Bart Miller: **bart@cs.wisc.edu**

Dorian Arnold: **darnold@cs.unm.edu**

Mike Brim: **brimmj@ornl.gov**

Phil Roth: **rothpc@ornl.gov**

Ben Welton: **welton@cs.wisc.edu**

Bill Williams: **bill@cs.wisc.edu**

http://www.paradyn.org/mrnet/