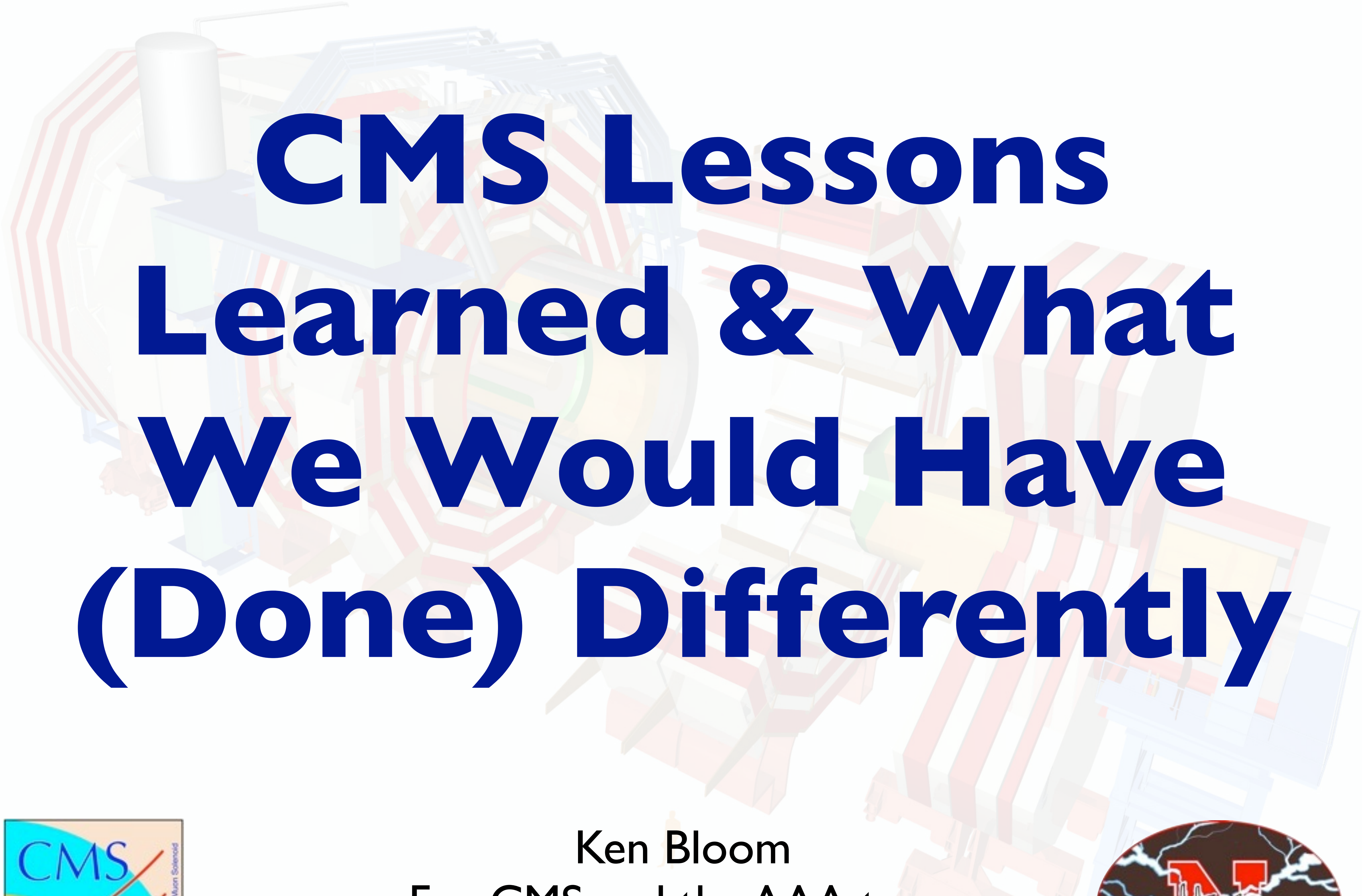


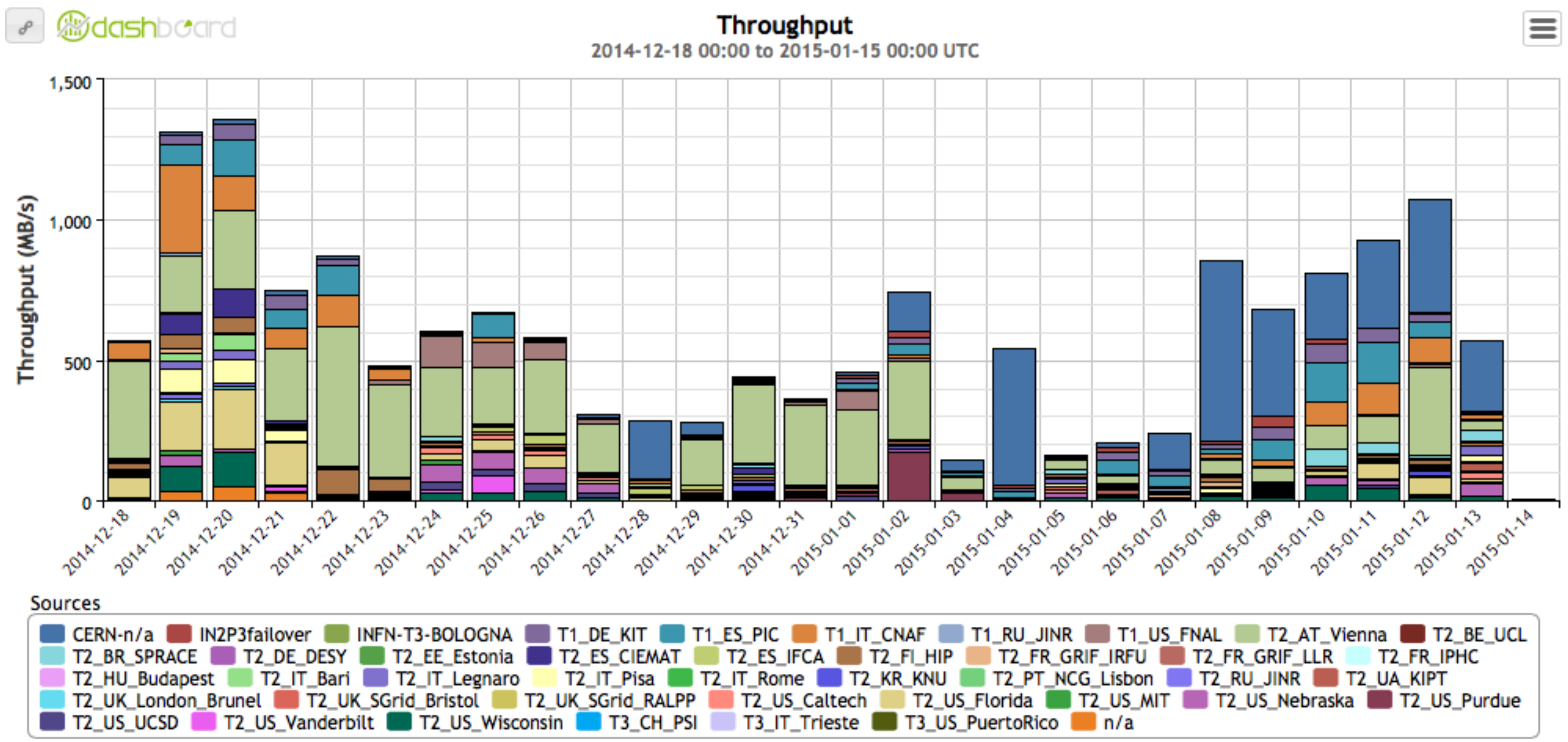
CMS Lessons Learned & What We Would Have (Done) Differently

Ken Bloom
For CMS and the AAA team
January 27, 2015

Lesson #1: this works!

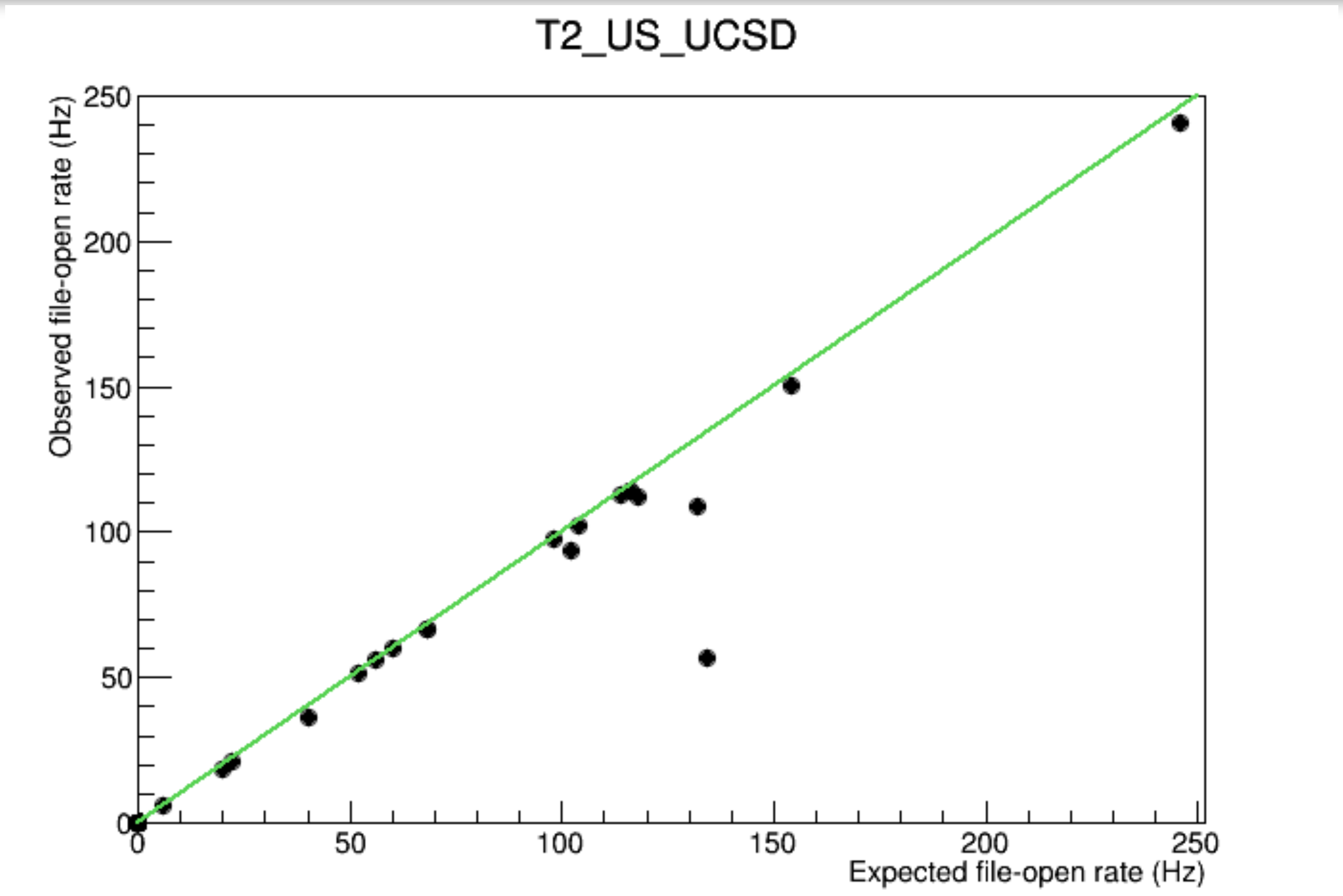
- Any Data, Anytime, Anywhere (CMS implementation of xrootd) has been enthusiastically received and implemented in CMS, by both sites and users, and is a key piece of the Run 2 computing strategy
- A good fit for CMS:
  - File namespace and I/O model turned out to be an excellent fit
  - Effort made to optimize WAN reads made AAA useable and was beneficial for CMS as a whole
  - Was easy to implement within CMS system; many applications simply enabled via fallback mechanism which requires only three lines of configuration
- AAA is everywhere:
  - Data available from all T1 sites, all but three functional T2 sites
  - Access via fallback mechanism available ~everywhere

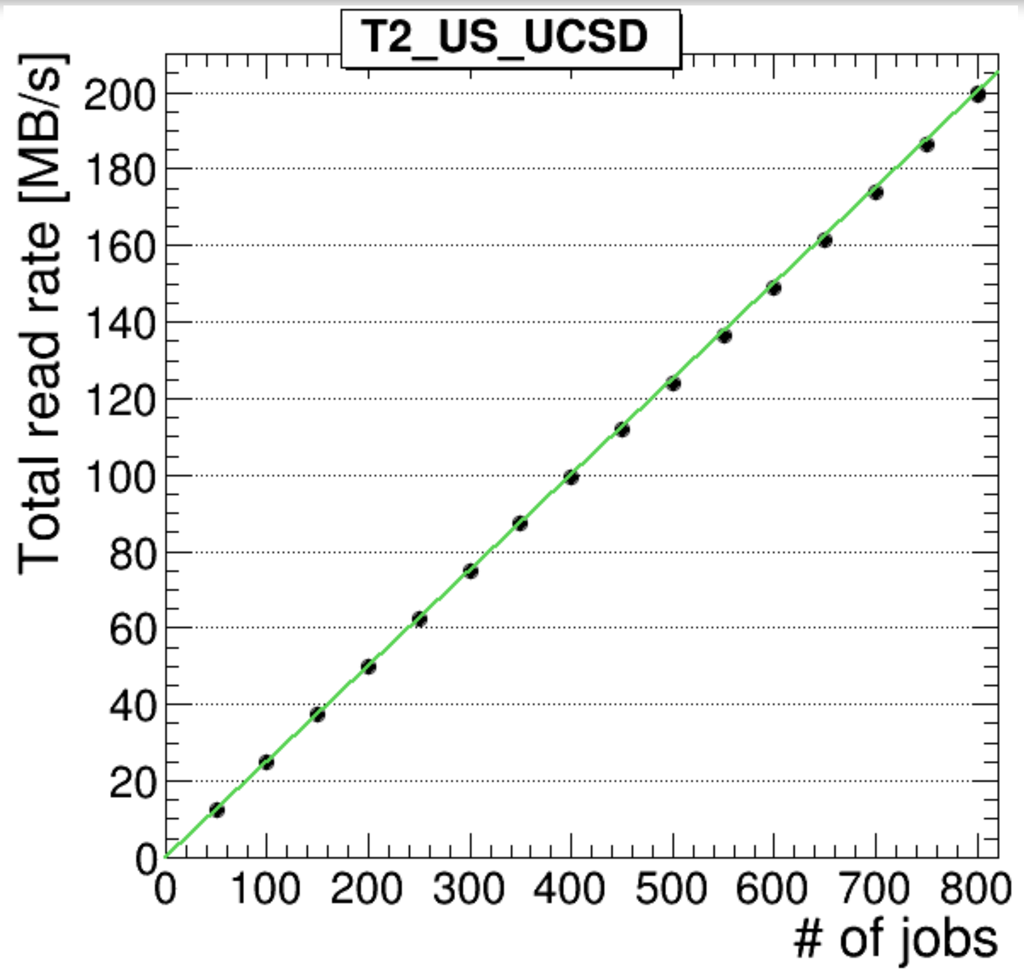
Throughput



- Sometimes have in excess of 1 GB/s moving via AAA
  - Average transfer rate in PhEDEx 0.5 GB/s during this time, comparable
- NB: this tally is incomplete, e.g. missing most of FNAL! (more later)

Scale testing





- Probed performance of ~35 sites with tests of file-opening and file-reading rates
  - Varied performance, but ~20 sites can successfully handle 600 simultaneous open connections, reading total 1.2 Gbit/s
- Also have performed system-wide tests of simulated loads; observe little lost processing time from job failures

Happy users

- Greater awareness of AAA thanks to last summer's CSA14 exercise, in which expansion of AAA use was a goal
- Real quotes from CMS members (not affiliated with AAA):
  - "It's like a dream come true…."
  - "These days I always run relying on AAA to serve data remotely, so there is no worry where the dataset is. Just need to set ignoreLocality to True in crab3 config."
  - "Xrootd is a really powerful tool that is going to make doing analysis a lot easier."
  - "AAA is awesome!"

Lesson #2: you can never have too much monitoring

- New technologies inspire a lot of curiosity about performance
  - Much more curiosity than exists for default technologies
- There has been a lot of demand for "monitoring"
  - Sometimes "monitoring" really means "accounting"
- But we struggled to define the right metrics to track
  - Amount of data flowing in/out of sites? # of successful file opens? # of jobs using AAA? Rescued by AAA? Increase in user happiness? Speed of analysis completion? # of emails in my inbox?
  - Providing a lot of data about the wrong information just adds noise
  - Different people want different metrics: whom to satisfy?
- Then, additional struggles to deploy the tools needed to get the metrics and to validate what was then being measured
  - Prettiness of dashboard makes people think it's truthful, but GIGO

Lesson #3: N goes in the exponent

- Lesson already learned from ~a decade of working with ~50 T2 sites in 26 countries: it's hard to get them all to do something
  - Particularly when it is something that's for a single VO
- Sites had to be encouraged one by one to deploy AAA
- Big struggle to get sites to deploy the monitoring tools
  - Made more difficult by the heterogeneity of the tools for different storage systems, and lack of support from some storage developers
- In general, a lot of the responsibility for configuration falls on sites; we can only plead with them to do the right things
- Q: Why hasn't WLCG embraced this more strongly and backed us up with the sites? Why can't we package this better such that it can serve all VO's in a similar way?

Lesson #4: the playing field is not level

- In an idealized implementation, if a file is available at N locations, it's OK to read the file from any of the N
- But in fact not all sites are provisioned equal
  - Storage responsiveness, WAN bandwidth…
  - Want to give users the best performance while also making the data federation as large/broad as possible
  - And perhaps want to protect against poor performance in real time
- Solutions are emerging for this:
  - Ability to separate a federation into "production" and "transitional" sites is available in Xrootd 4.1; try to get files from production sites first then fall back to transitional sites
  - Multisource routing, fallback to fallback part of 2015 analysis release

Lesson #5: fear of users and usage

- (Or, how is AAA like Obamacare?)
- Concerns exist that users could essentially perform a DOS attack on individual sites, or perhaps the entire system
- In working experience so far, such incidents have turned out to be rare, contained and unintentional!
  - "I trust AAA so much that I expect any failures are transient, so I just put in automatic retries of my jobs when they fail…."
- But it is a valid issue for individual sites:
  - When storage is accessed directly through local CPU's, required storage performance is determined by the number of batch slots
  - When storage is accessed remotely, sites have no control
- Sites do need something that will let them protect themselves if necessary ("throttles")
  - But how to make sure sites use them wisely?

Lesson #5: fear of users and usage

- Robust debate in CMS on how best to put this powerful technology to use for the maximal benefit of users
- Let users choose whether to allow remote access?
  - Give users maximal control over how they get their work done
  - Potentially maximally efficient use of CPU resources
  - "But we can't have everyone doing this!" — no regulation
- Only allow remote access as a last resort?
  - Jobs run where the data lives, only go to federation when in trouble
  - Probably don't get all possible benefits of AAA
- Make central decisions about remote access?
  - Implemented via Condor job overflows, not available everywhere
  - Could work if system is sufficiently responsive
  - Users don't always like having decisions made for them

Lesson #6: one piece of the puzzle

- More about how we pitch AAA, rather than AAA itself
- AAA itself shouldn't be regarded as a magic bullet for computing
- Sometimes it won't work right, but that's OK if it is part of a robust, resilient computing environment:
  - Worried about file-open failures? Have automatic job resubmission.
  - Worried about too many jobs trying to read popular data from a single site? Deploy popularity-based dataset distribution.
  - Worried about straining networks? Make more access local by reducing event sizes and allowing each site to host more events.
  - We are now doing all of these things!
- There are many components to CMS computing, and they support each other to give the best throughput and overall experience for the user

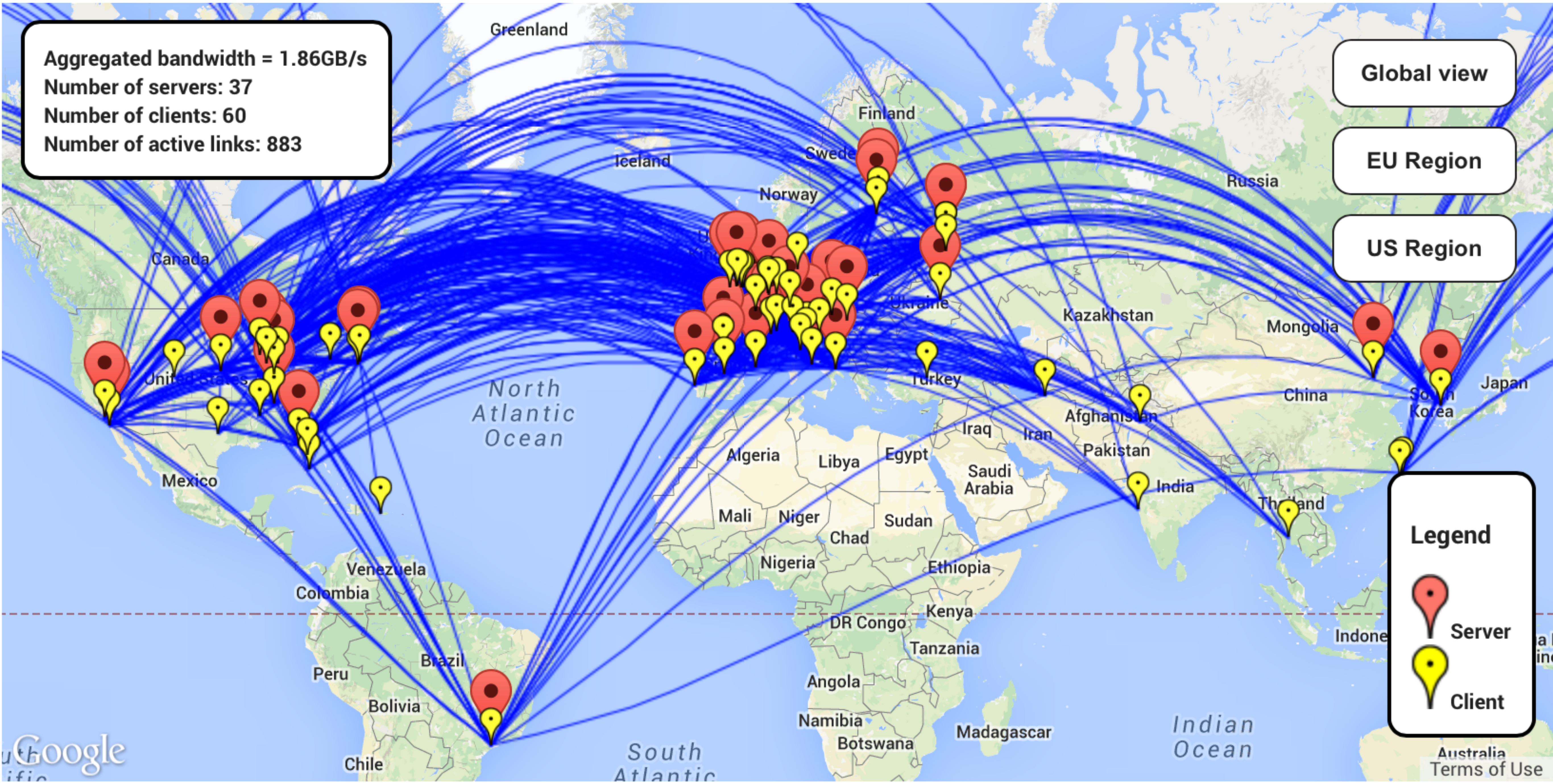
If we were to start again?

- If only we had had AAA from the very start of our planning!
  - Build it in as a fundamental piece of CMS computing, not an add-on, and use it to influence the entire computing model
  - Create an expectation among sites, experiments, WLCG that this is a fundamental service (like a CE or an SE) for LHC participation, and that sites should be provisioned appropriately
  - Then we could take maximal advantage of the technology
- Technical things that would be nice to have at the start:
  - Better understanding of what we want to monitor/account and how
  - More central configuration of site behavior
  - Management of heterogeneous site capabilities
  - (but now we know about these and are making progress)

Conclusion: lesson #1 redux

- This works!
  - The system can work at the necessary scale
  - We have a growing user base, and they give positive feedback
  - CMS has identified AAA as a key element of the Run 2 computing strategy, for both organized and chaotic workflows
  - All thanks to a lot of hard work from very many people

Any Data, Anytime, Anywhere!



Aggregated bandwidth = 1.86GB/s
Number of servers: 37
Number of clients: 60
Number of active links: 883