MathWorks: A Case Study in NAS

Ira Cooper

Senior Systems Software Engineer

Samba Team

Who is MathWorks?

- MathWorks develops MATLAB™ and Simulink™.
  - Including 80-90 Toolboxes!
- We are a company of ~2500 people (~1000 developers) across many sites:
  - United States – HQ
  - France
  - Germany
  - Japan
  - India

Who is MathWorks? 2

- MATLAB™ and Simulink have Toolboxes in:
  - Model Based Simulation, Design & Verification
  - Finance
  - Statistics
  - Embedded Code Generation
  - Symbolic Mathematics
  - Biology
  - Automotive Engineering
  - Aeronautical Engineering
  - And many more areas.

What makes us an interesting site?

- HPC Scale, but not typical HPC Style
  - Heterogeneous not Homogenous – NFS, SMB, SMB2.
    - Windows 7 32/64 Bit, Windows XP 32/64 Bit, Linux, OSX.
  - Small file I/O heavily metadata driven, instead of large block file I/O.
  - Build times for a sterile build can be 24hrs+.
  - Thousands of linear hours of tests that need to be run.
  - Thousands of cores of compute power and growing constantly.
  - Enterprise environment to integrate into.
  - Huge amounts of automation.
  - Heavily cross protocol.

Additional Challenges

- Third party products.
  - We act as a major integration point.
- Performance + Monitoring
  - Our speed is critical to the company’s success.
  - Performance that can’t be quantified is not useful.
- High quality and performance requirements.
  - A 1 in 1000 fault will be seen several times a week in our environment, potentially.

Major Design Factors

- High Reliability:
  - Must be able to support our 24hr+ builds.
- Performance:
  - Over 100k+ mixed ops per fileserver is a start.
- Cost:
  - F1 car, for the cost of a Scooter.
  - We can’t have our storage costs “out of control”.

Other Design Factors

- Introspection
  - The ability to work with the server to determine where a given issue is.
- Bug Fixing / Verification
  - 90% of the problem is usually finding the bug.
  - Alas the other 90% is convincing a vendor to fix it.
    - On their schedule
      With their priorities

Design Decisions

- High Availability
  - Do we really need HA running a batch system?

- “Deal with the Devil”
  - Each server will go down one day a year, and we can’t say which.
  - But the system will go twice as fast and cost much less.
  - For a batch system, this is a can be a good tradeoff!
    - If your users buy in.

Design Decisions 2

- Simplicity
  - This allowed us to go from proof of concept to production in 6-8 months.

- Timely support is critical
  - Support can be provided in house.
  - We know and understand the priority of our own issues better than any vendor can.

Design Decisions 3

- Open Source!
  - Introspection via reading the code is hard to beat.
  - The ability to directly collaborate with our “upstream vendors” at a code level really simplifies things.
- ZFS
  - RAID – RaidZ, RaidZ2, Mirror
  - Snapshots
  - SSD Read Cache
    - Not really tiering, but close enough
  - SSD Write Cache
    - Required due to the heavy NFS traffic. Synchronous write performance matches our SSDs, max IOPS

Design Decisions 4

- OpenSolaris/Illumos.
  - Very reliable.
  - dtrace and other analytical tools have proven very valuable over time
  - The best Open Source platform for ZFS.

Design Decisions 5: Why Samba?

- We couldn’t use Solaris Kernel CIFS when we started.
  - Also no SMB2.

- Likewise seemed to be missing notify, which is a key feature for us.

- Samba has a very “mature” codebase.
- Samba has a very strong community.
  - This leads to more features and bug fixes!

Starting Hardware

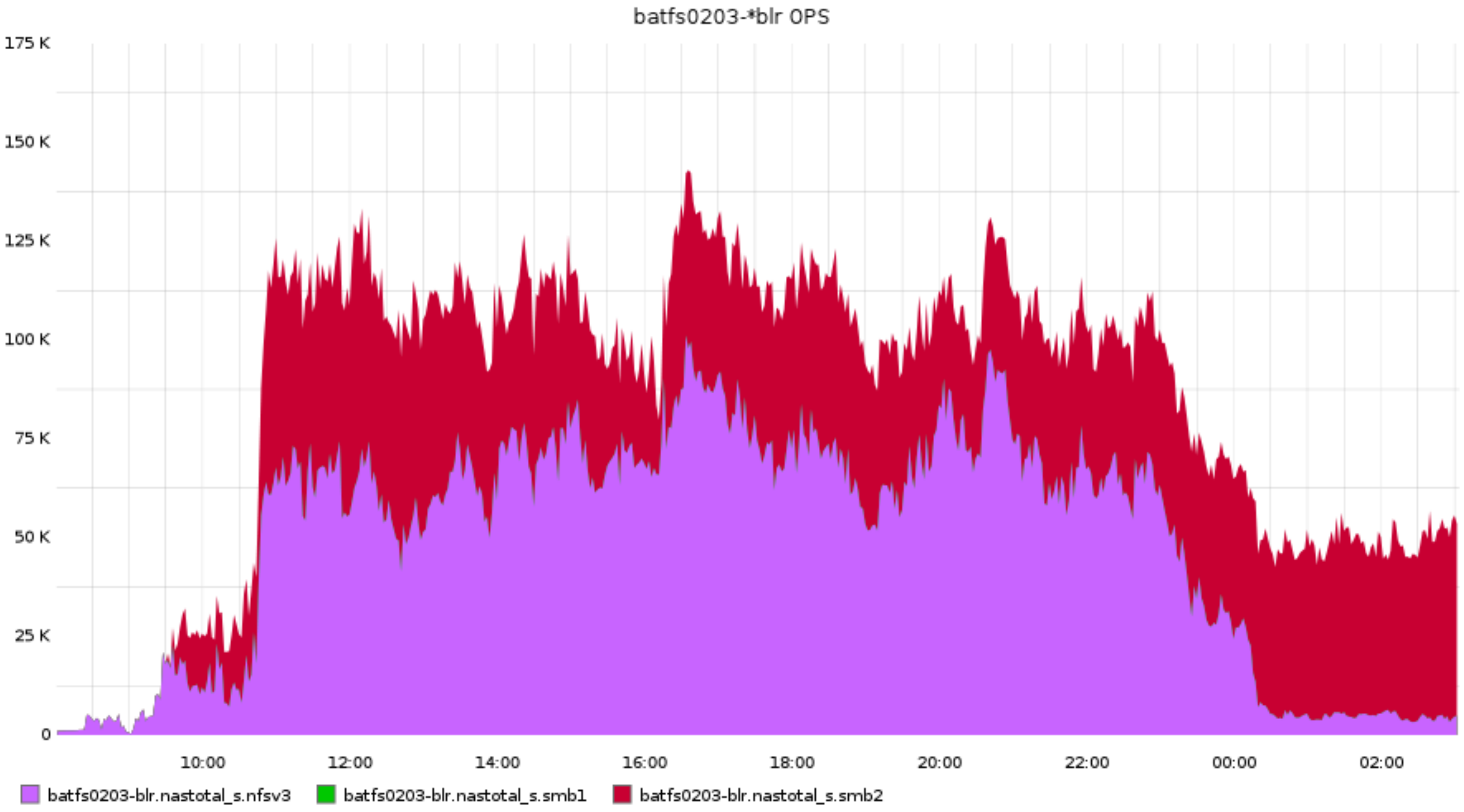
- NexentaCore + Samba 3.6-GIT
- SuperMicro Servers
- We started at:
  - 72GB RAM
  - 24 Core – Westmere
  - 3 L2ARC SSDs
  - 2 ZIL SSDs
  - 19 10k drives
  - All in 2U

Hardware Today

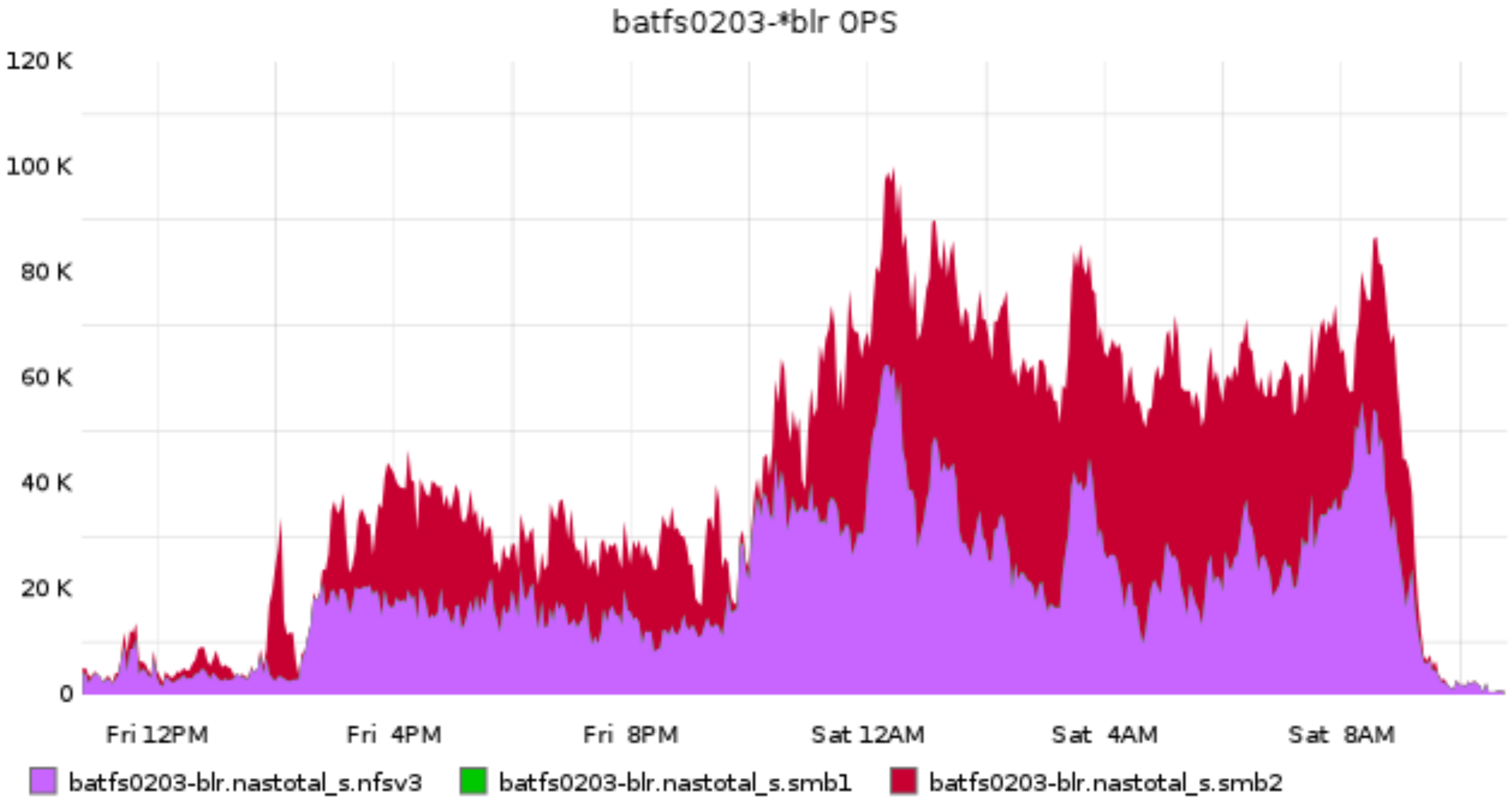
- Current specs:
  - 192GB RAM
  - 24 SSDs
  - 24 Core – Westmere

- We are also building HA servers.
  - Head nodes have:
    - 192GB RAM
    - 24 Core – Westmere
    - 6 SSDs
  - Trays:
    - 2 ZIL
    - 22 SAS 7200 RPM SAS drives

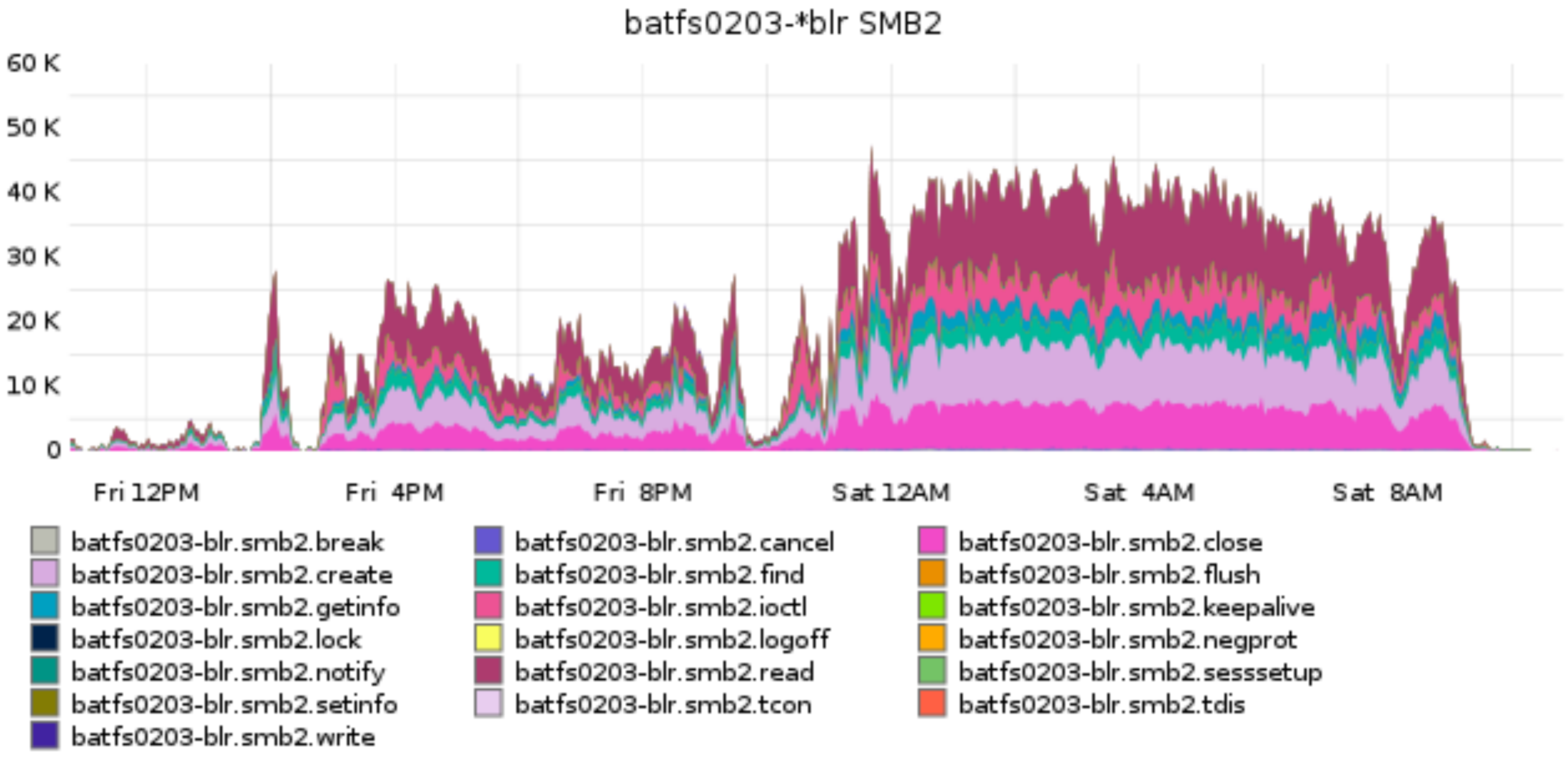
Results – Pre-production Qualification



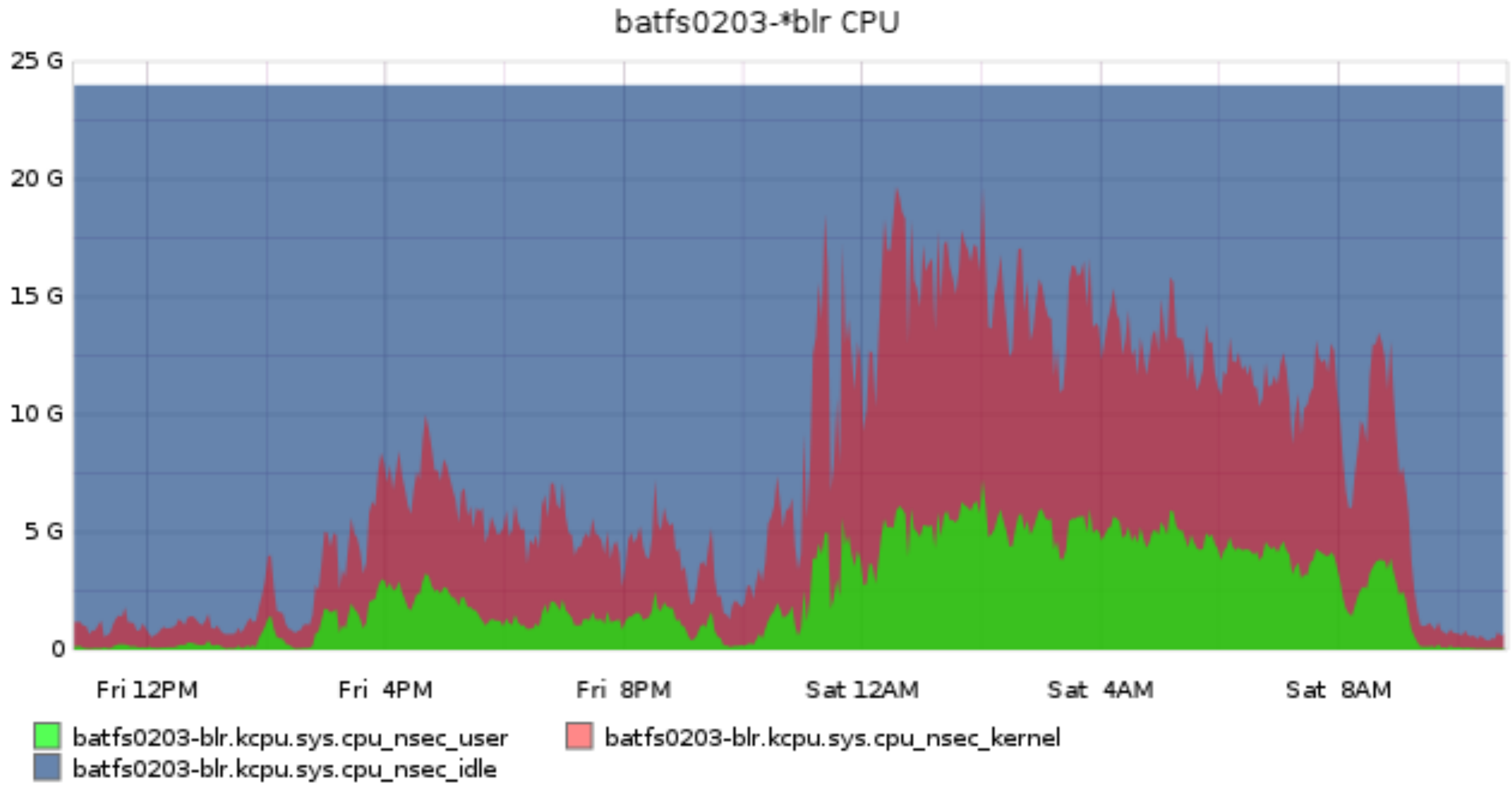
Production: Overall Server OPS
April 27-28, 2012



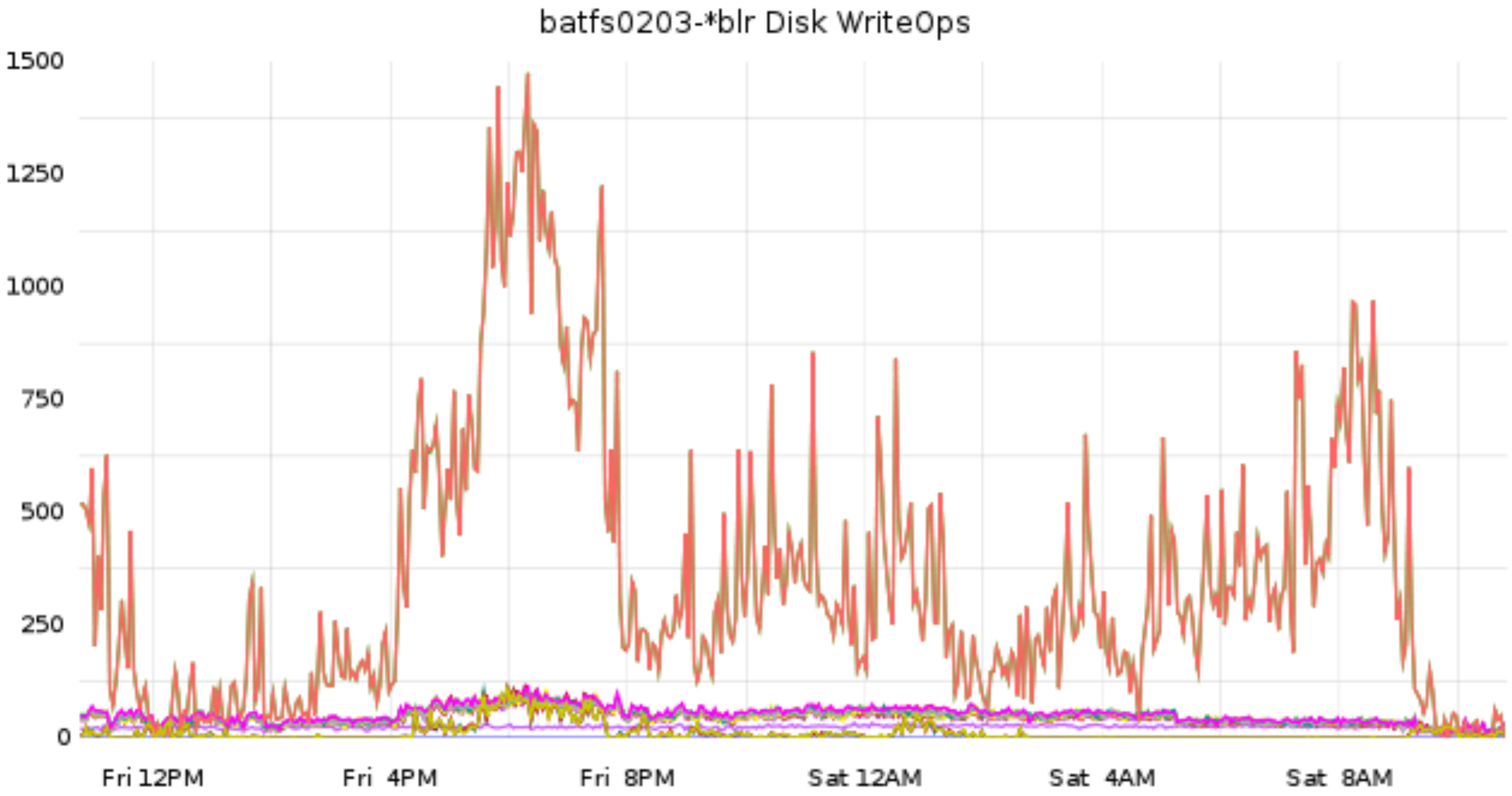
Production: SMB2 OPS Break Down April 27-28, 2012



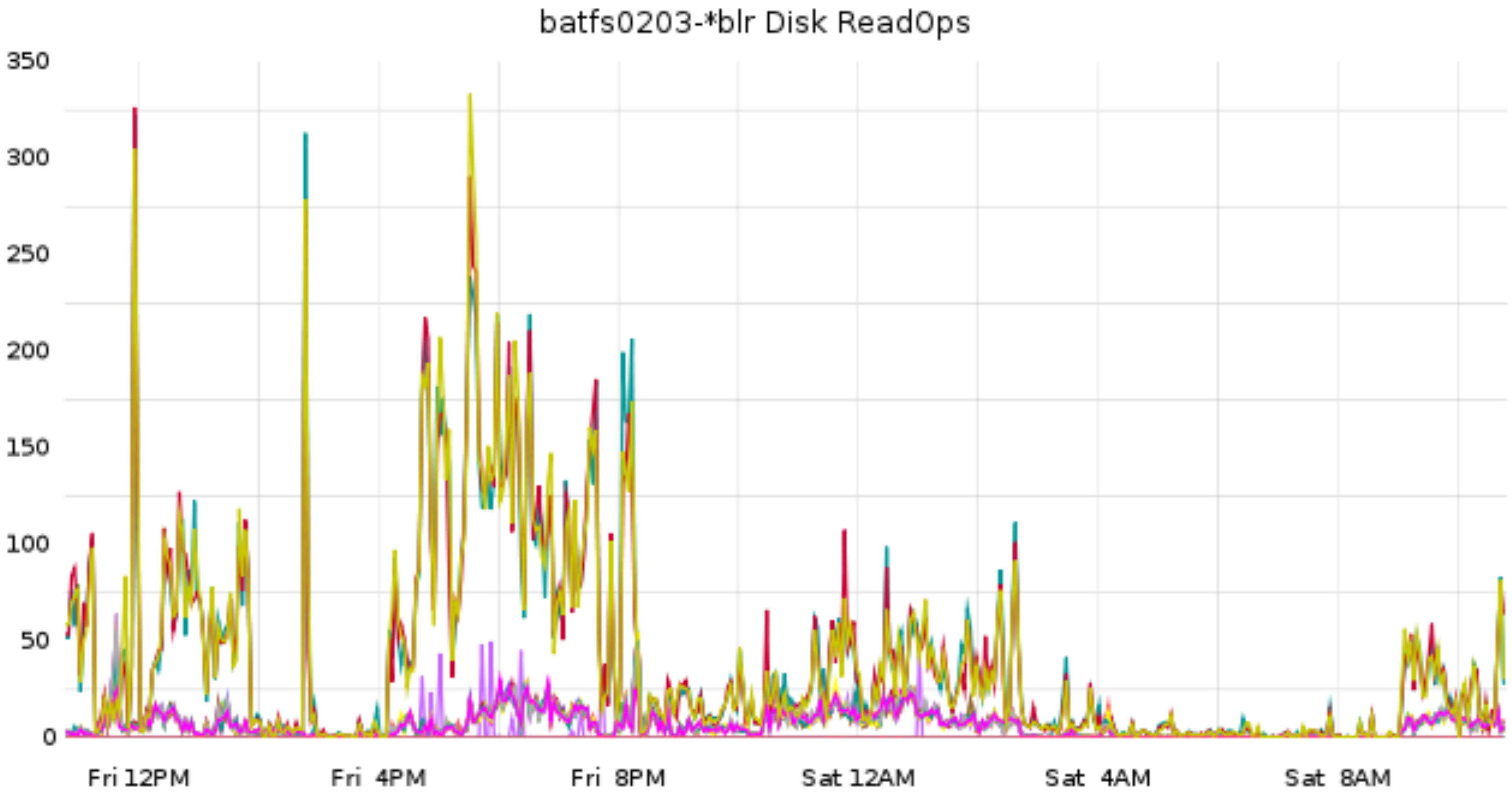
Production: CPU Use
April 27-28, 2012



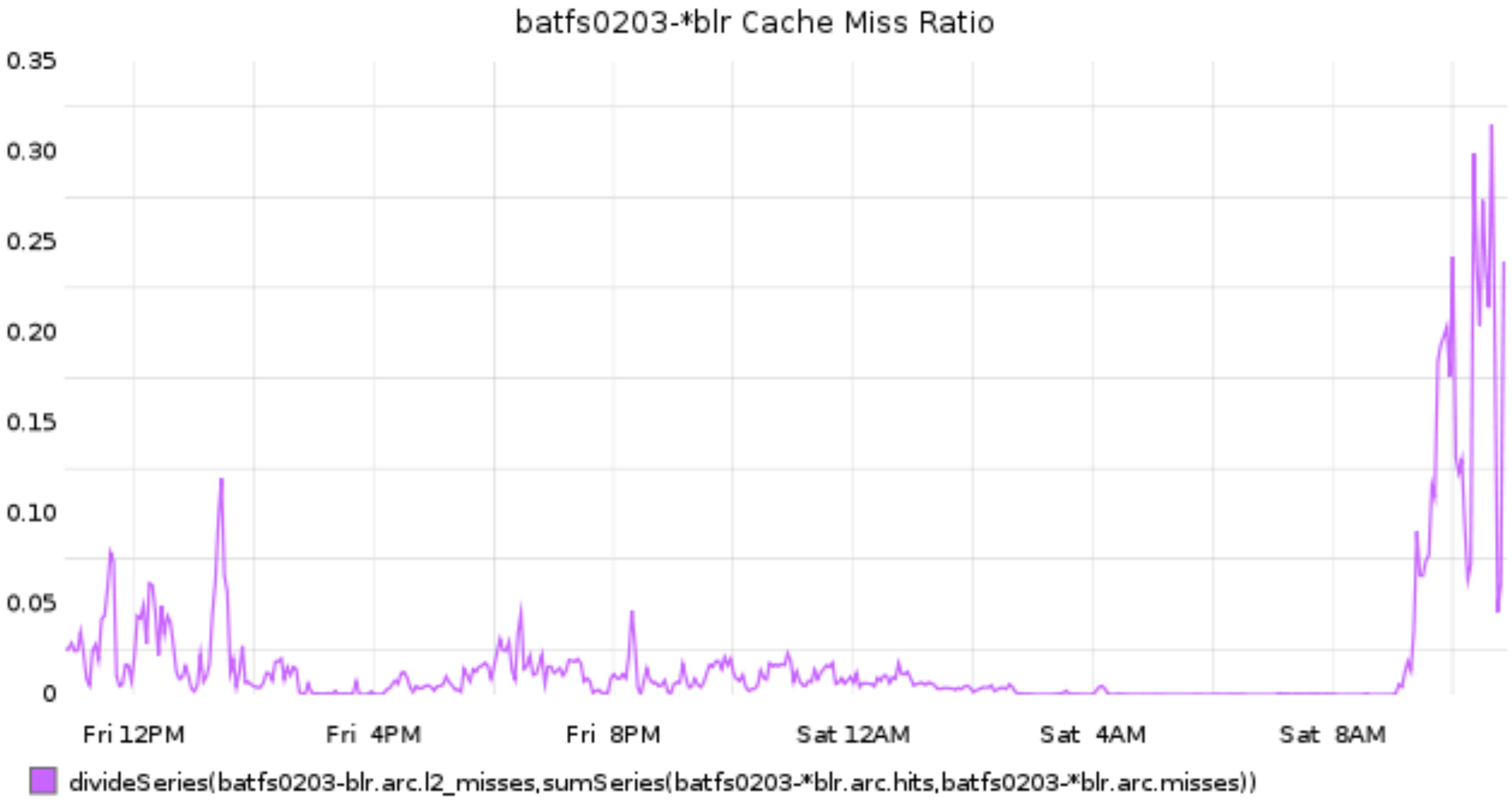
Production: Write OPS, Backend
April 27-28, 2012



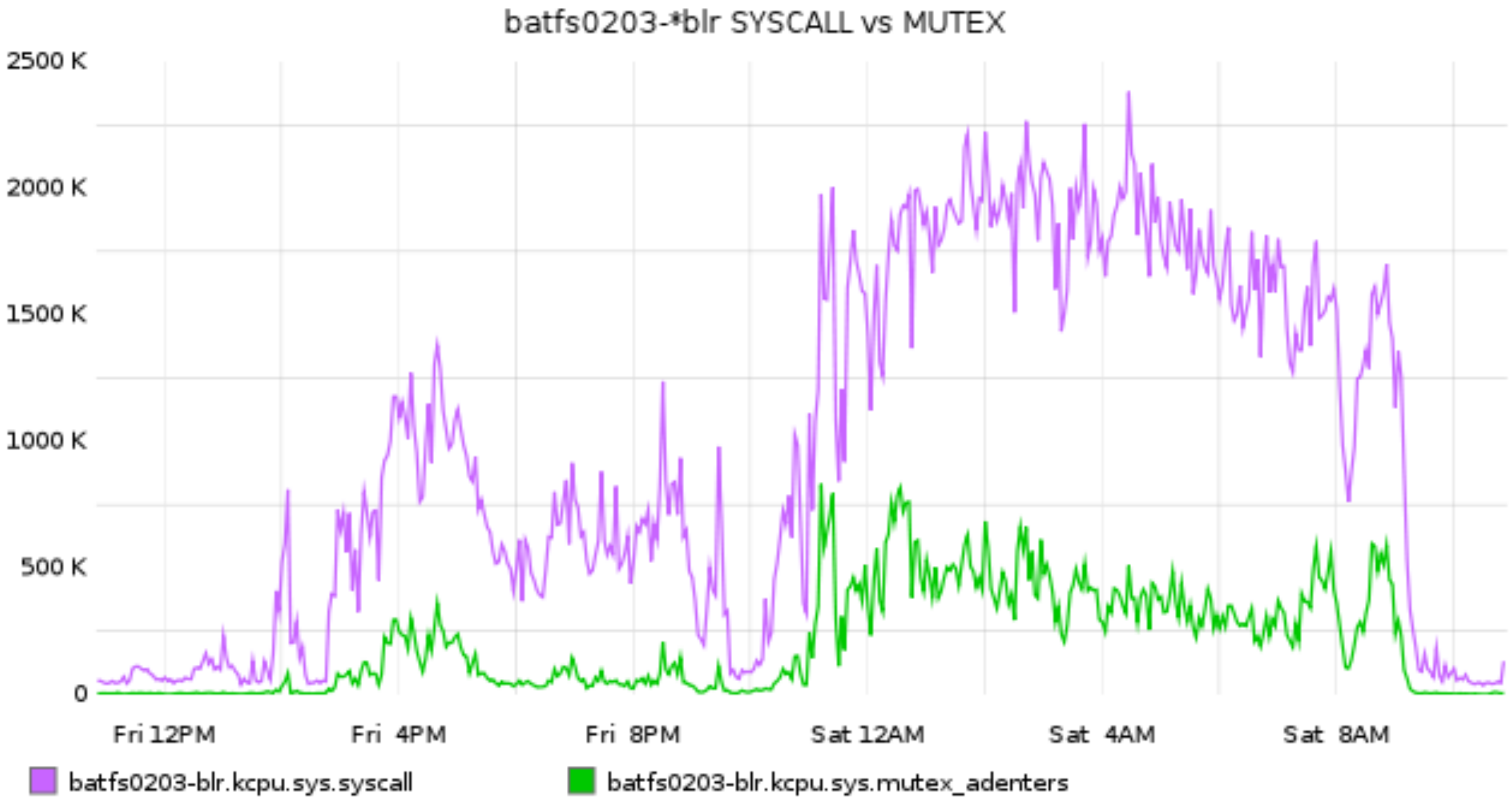
Production: Read OPS, Backend April 27-28, 2012



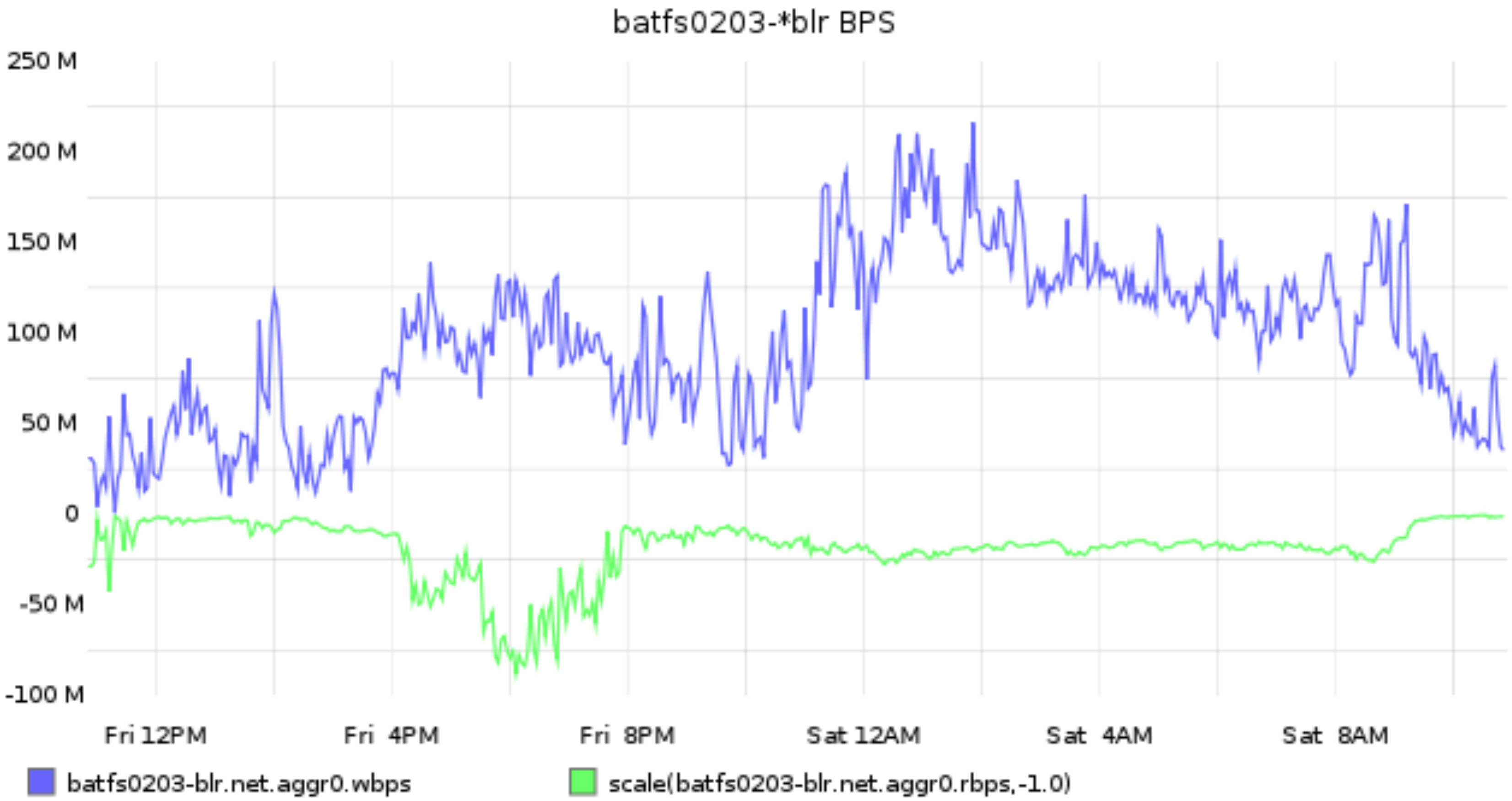
Production: Cache Miss Ratio
April 27-28, 2012



Production: System Call vs. Mutex Lock Miss April 27-28, 2012



Production: Bandwidth – Bytes per Second April 27-28, 2012



Results – Good!

- Overall, the project is a major success story.
- Management is very understanding of the effort involved, because we are so hands on.
- When there are problems in the lab, management wants them to be on the server side!
  - Because they know we can fix it!

Results - Bad

- SMB 2.0 Issues.
- Solaris/OpenSolaris Platform issues.
- Pure Samba issues.
- Note: We'd expect issues with any platform we bring in.

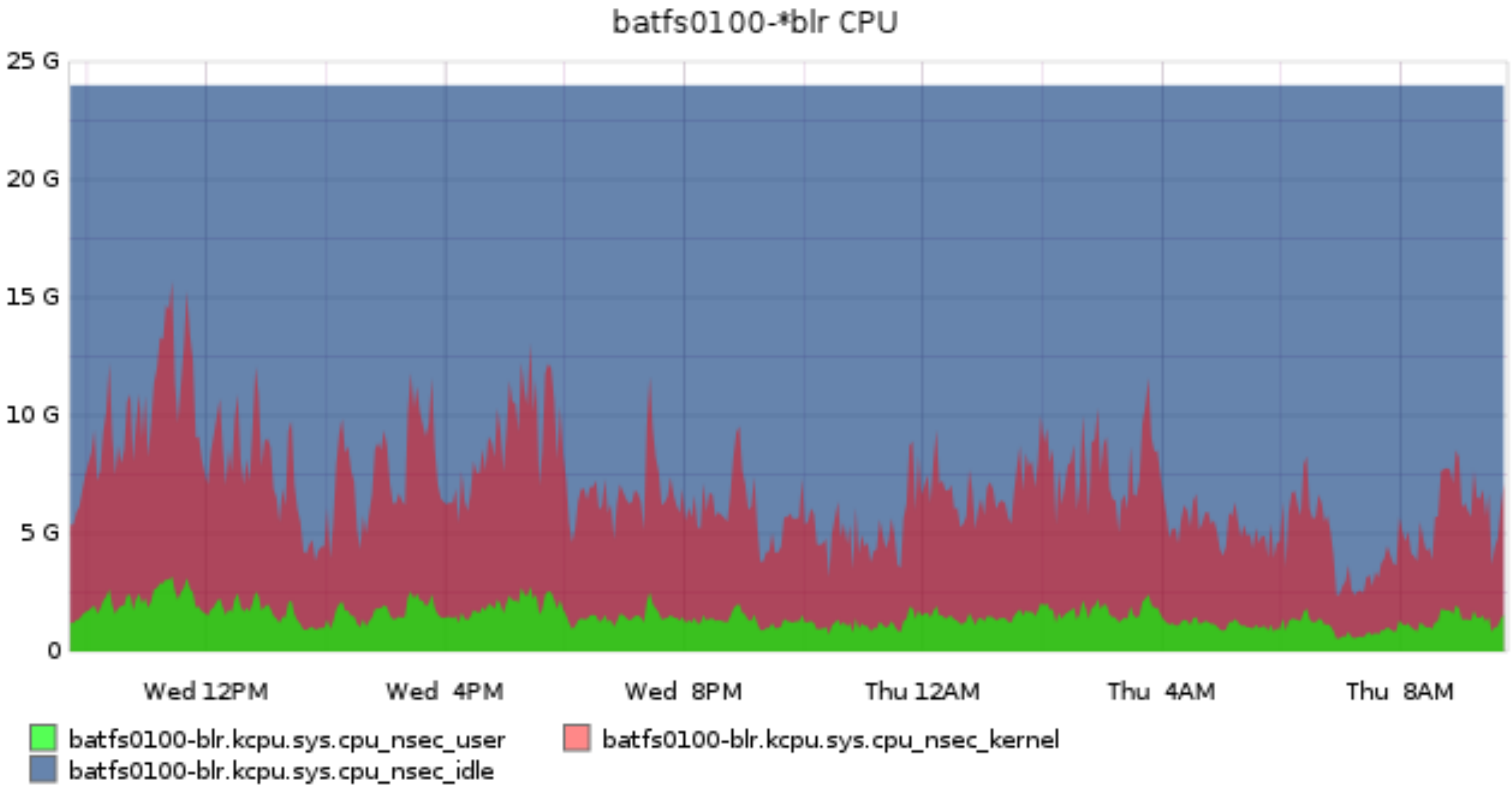
Security Issue: DOS on Samba

- This is the issue that caused the release of 3.6.3.
- Pre-production testing showed a large spike in CPU activity.
- I'll lead you through how we found the issue.
- Credits to:
  - Youzhong Yang – MathWorks
  - Jeremy Allison – Google/Samba Team

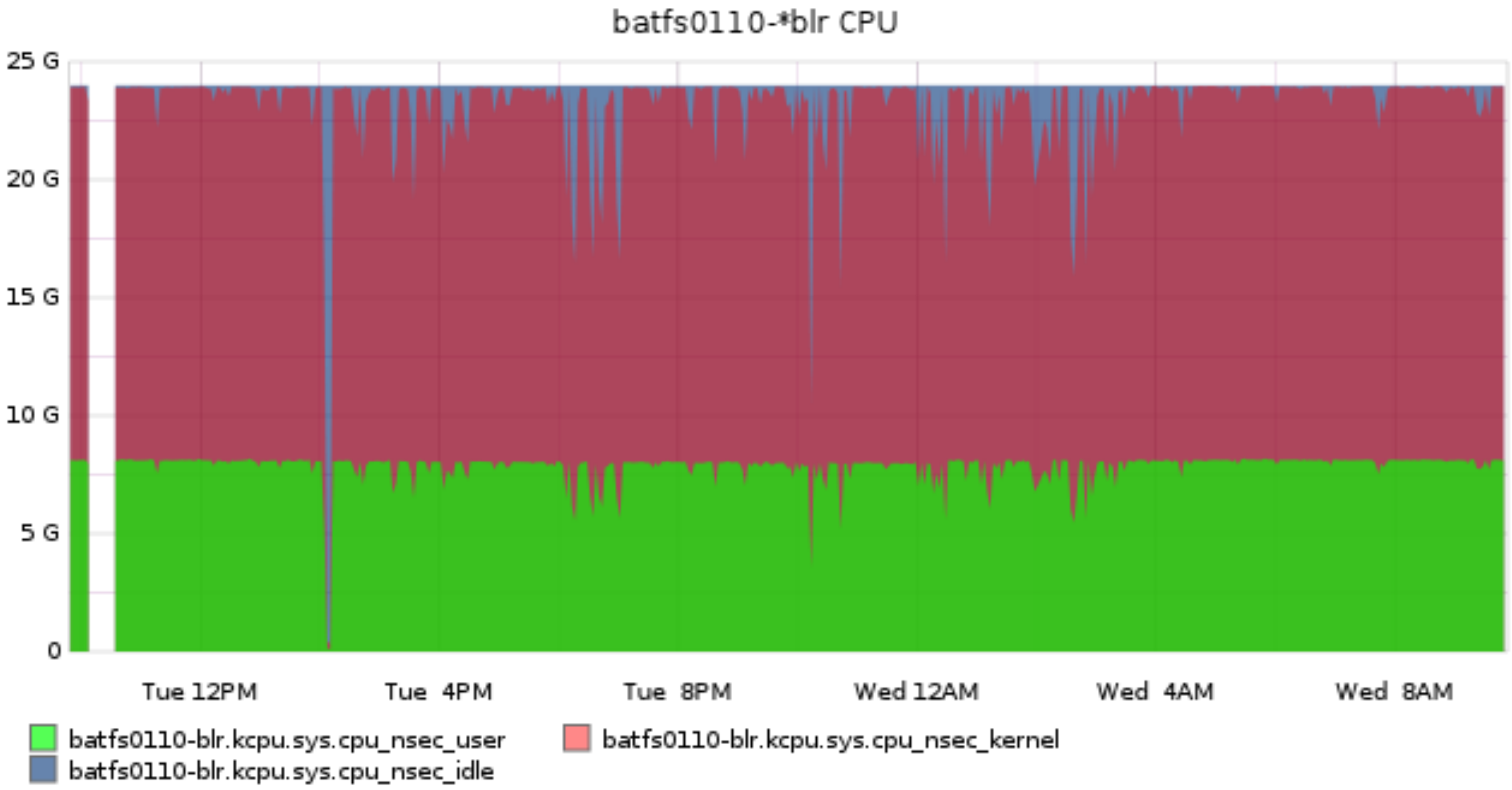
Initial Problem:

- The new Samba release is slow.
- It is pegging our CPUs.
- What's wrong?

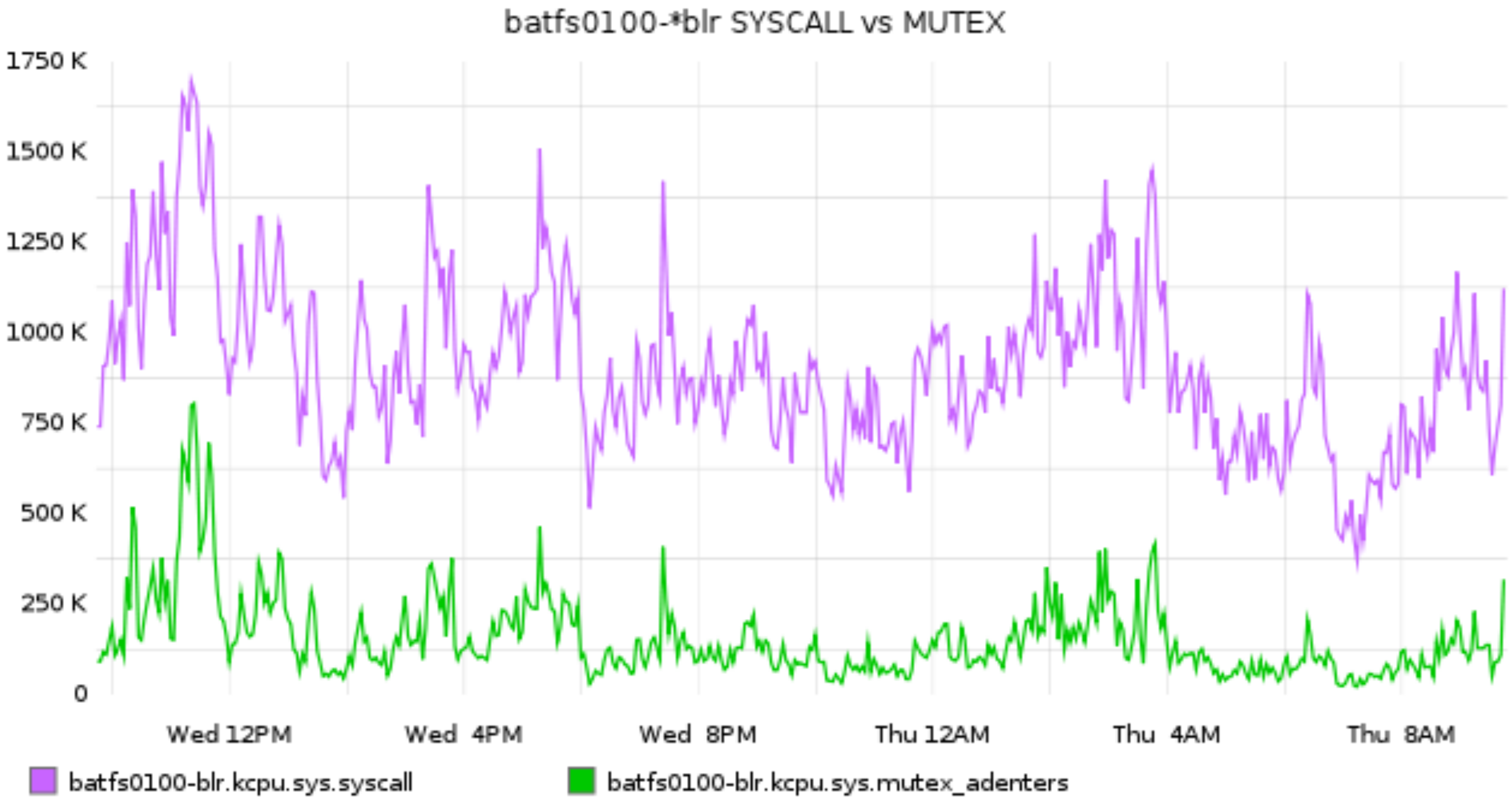
3.6-GIT: CPU Utilization



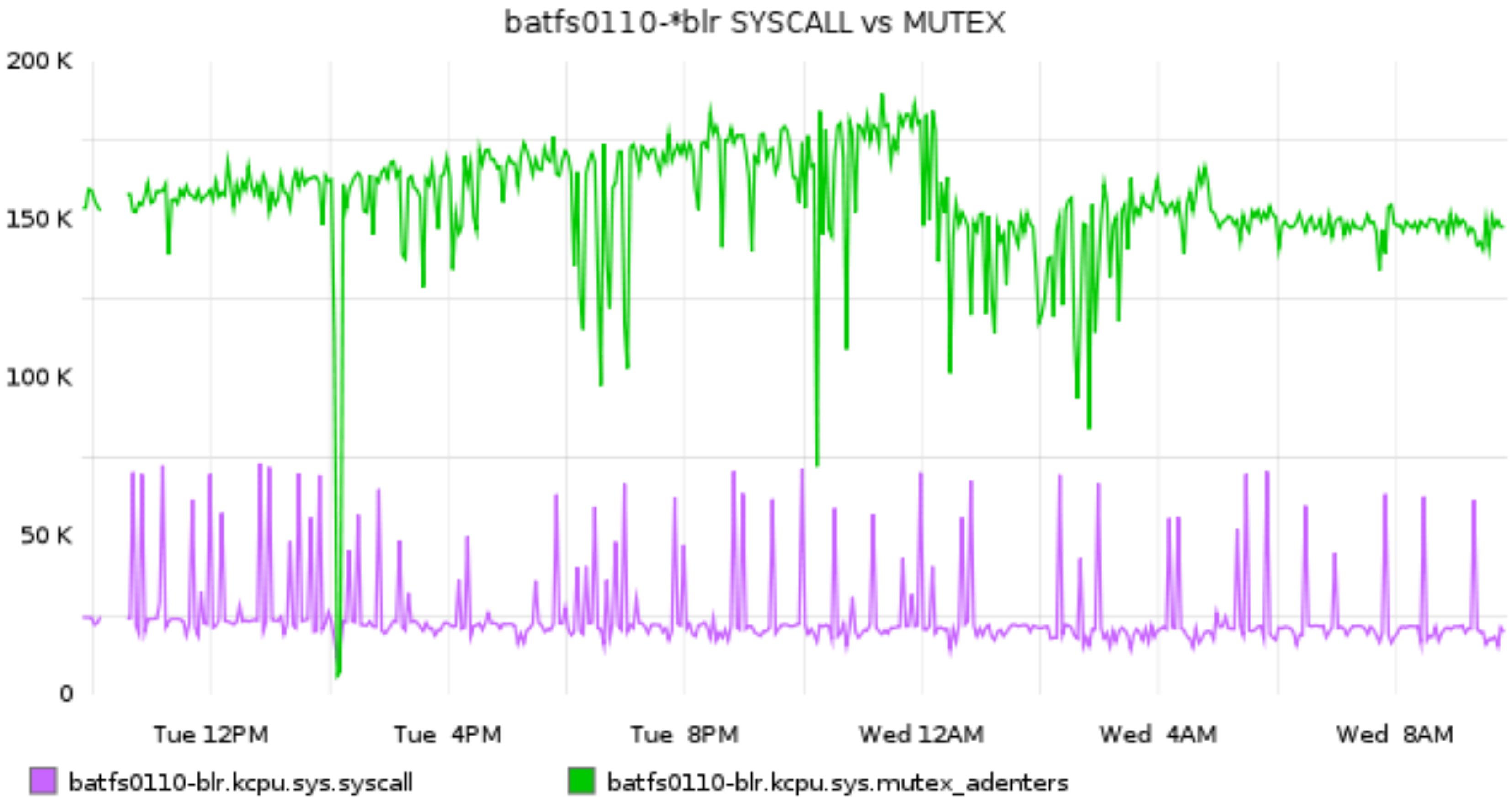
3.6.2-GIT: CPU Utilization



3.6-GIT: Syscall vs. Mutex



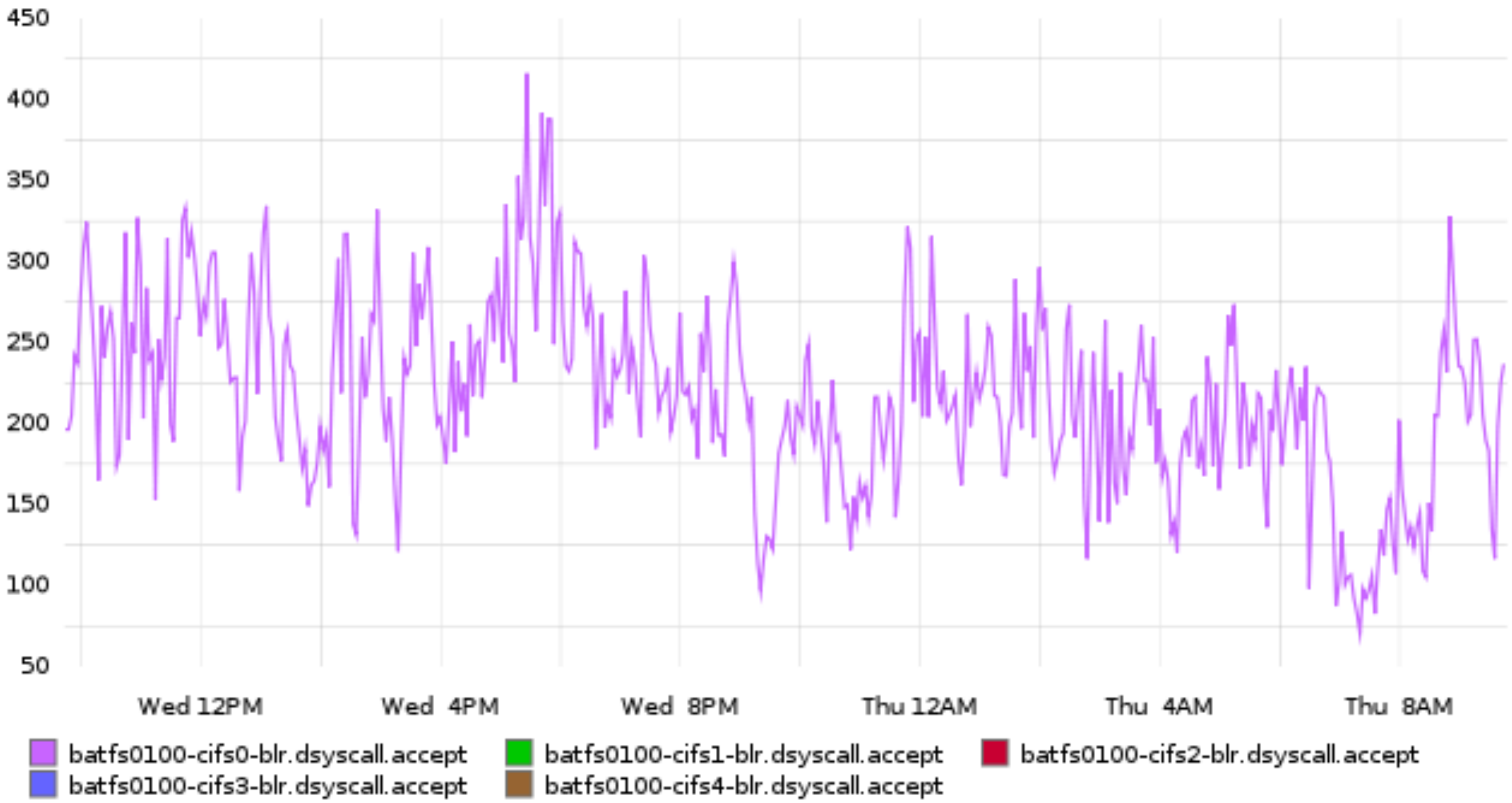
3.6.2-GIT: Syscall vs. Mutex



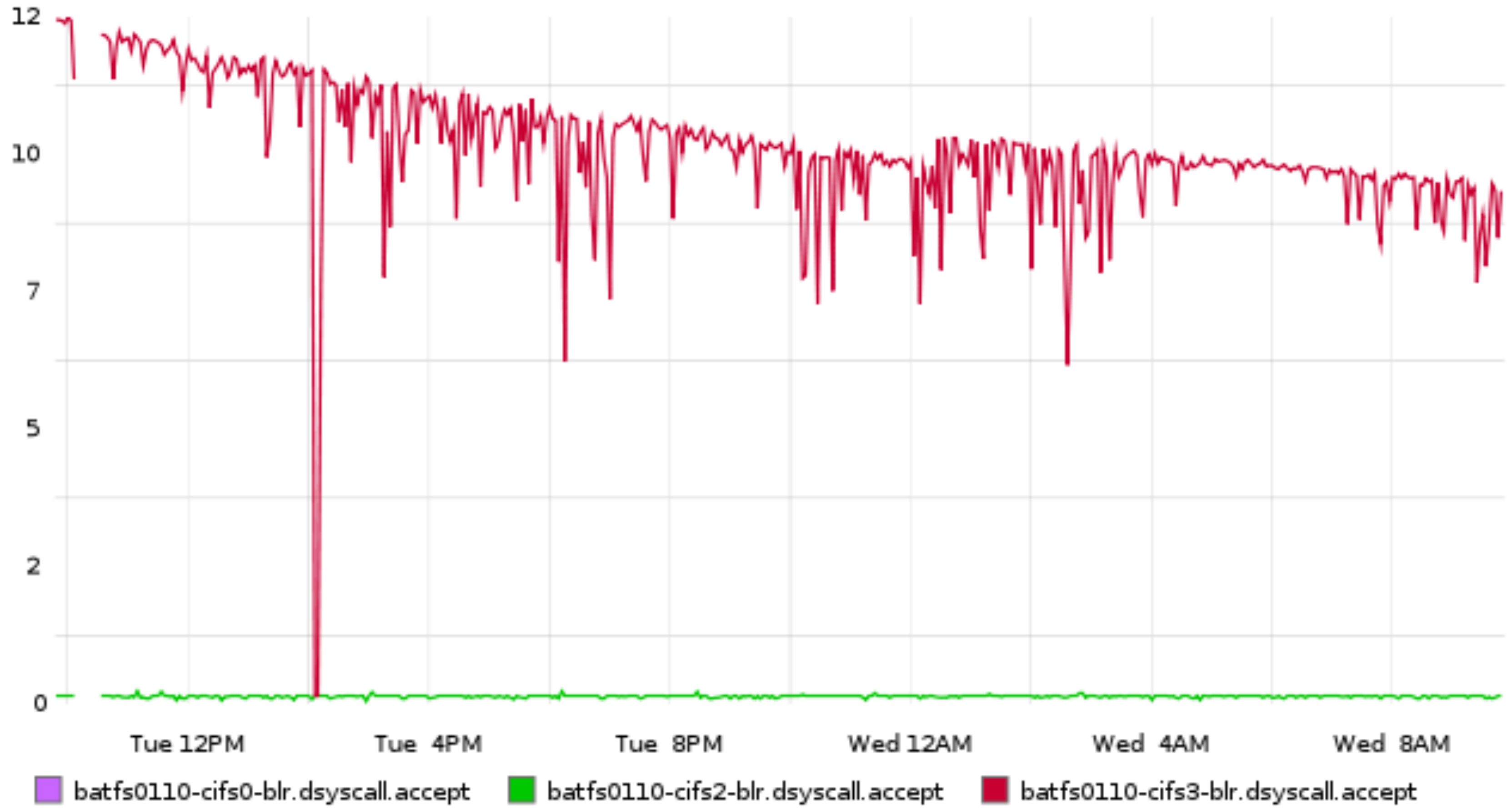
What’s wrong?

- Something is “different.”
- It isn’t the environment.
- It must be the code.
- It was noticed that new connections to the server were taking too long.
  - We added a new metric!

3.6-GIT: Accept Calls per Second



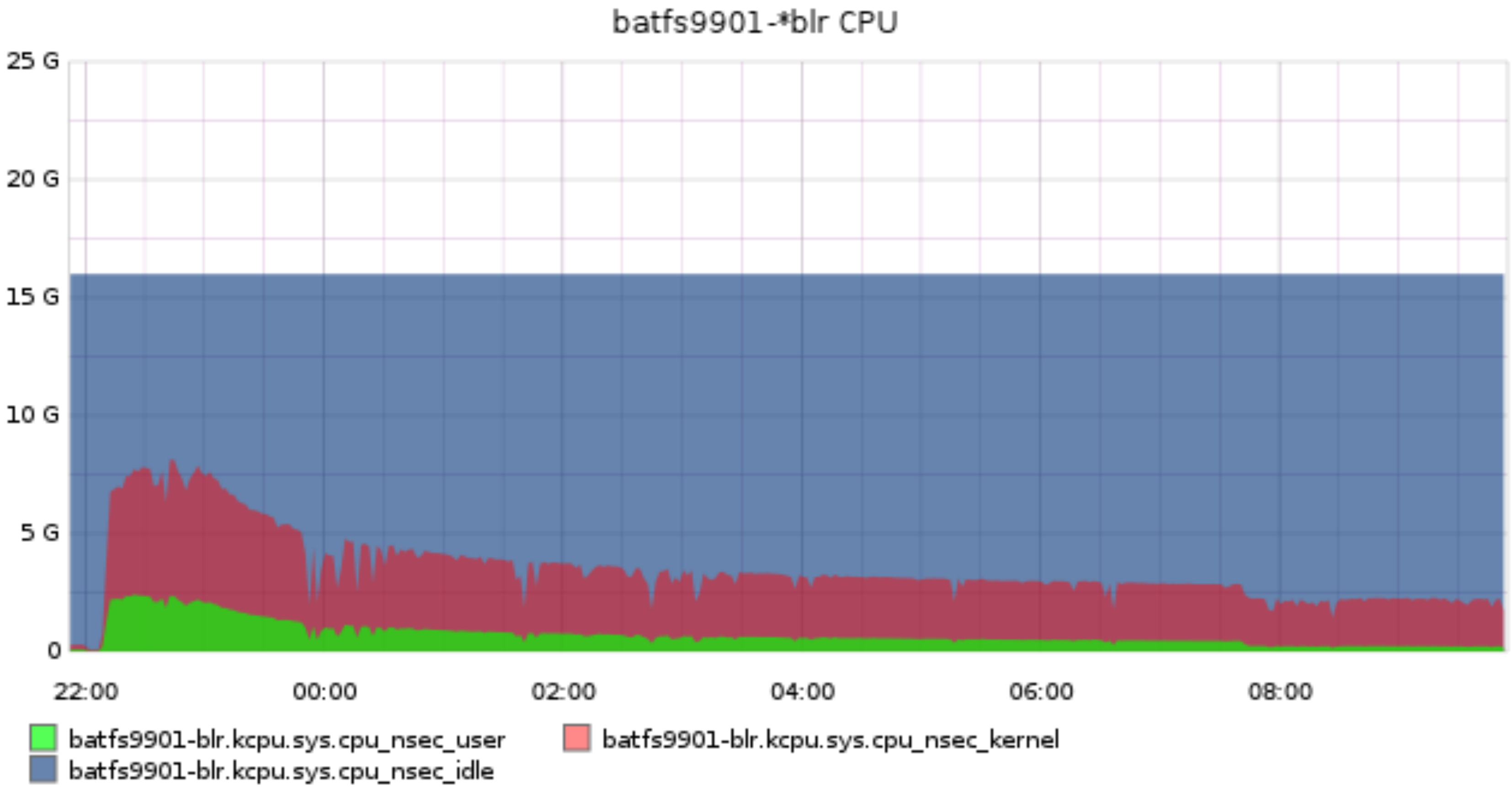
3.6.2-GIT: Accept Calls per Second



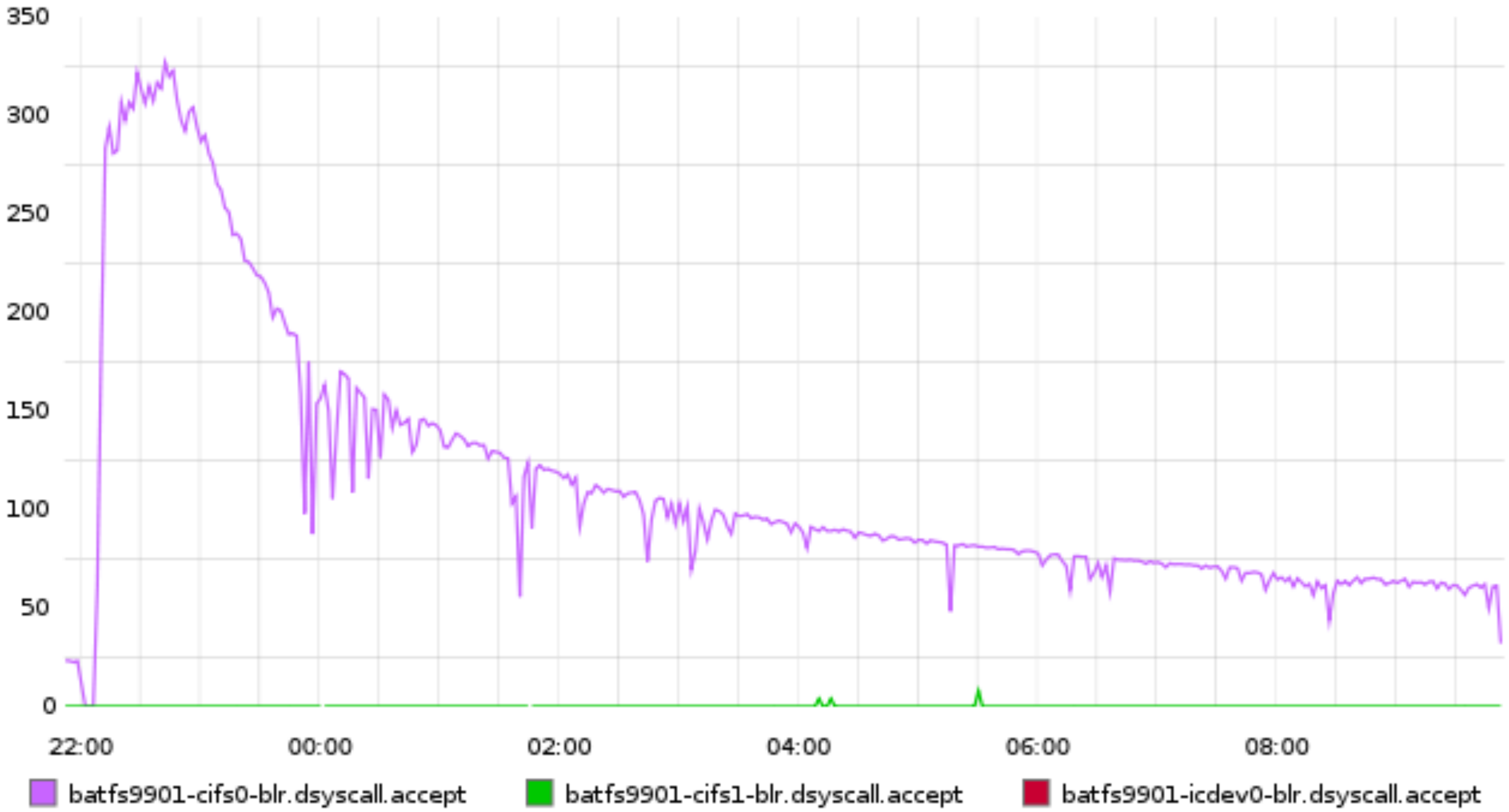
Characterized!

- Something is making smbd connect really slowly, and take up too much CPU.
- Test it!
- Small set of smbclients looping against a dev server:

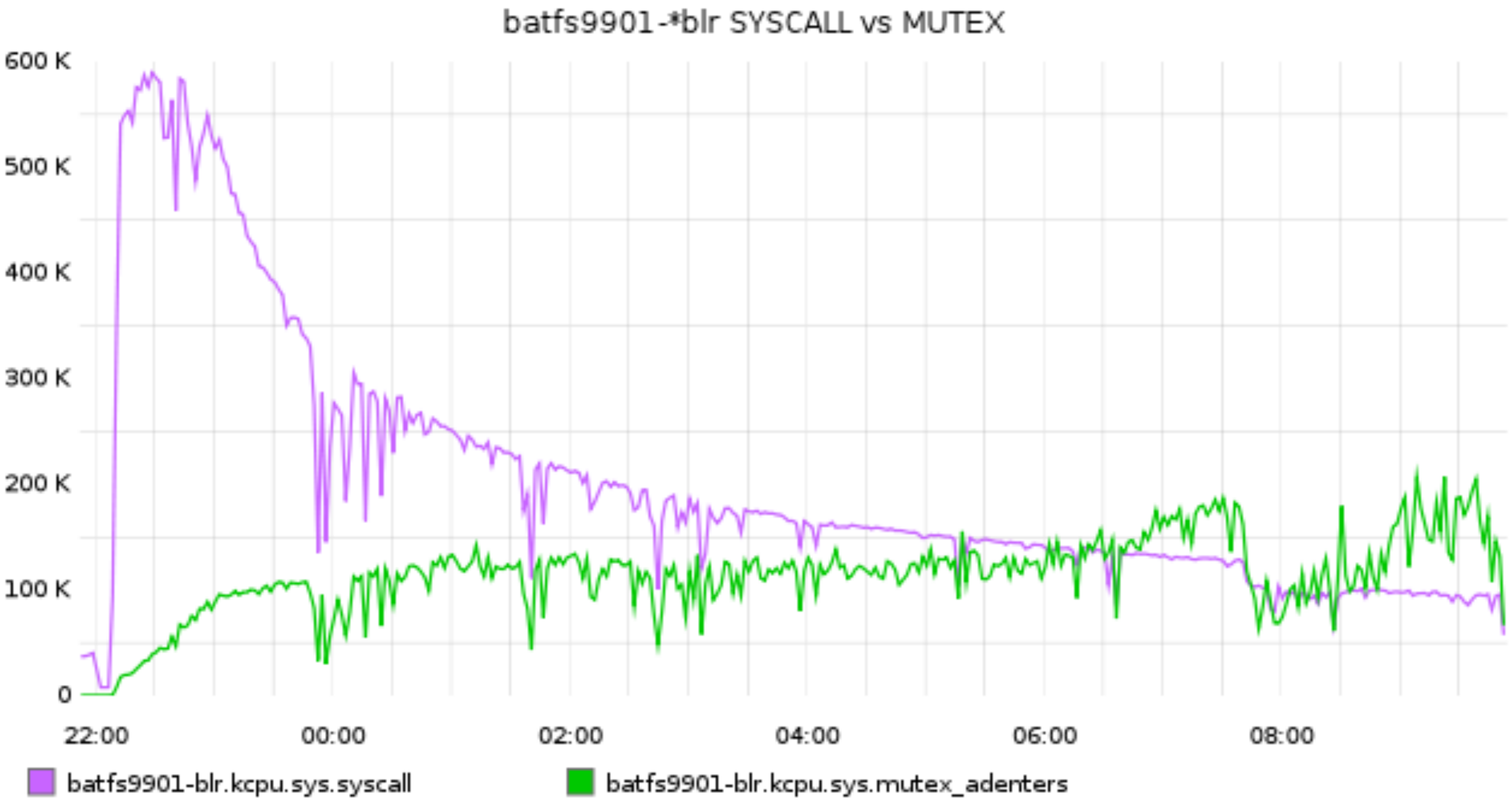
3.6.2-GIT in Dev: CPU Use Connect Testing



3.6.2-GIT in Dev: Accept Call Rate Connect Testing



3.6.2-GIT in Dev: Syscall Vs. Mutex Connect Testing



What Was This Regression?

- Clearly samba had never done this to us before.
  - We had the data to be very confident this was a regression.
- Bisect?
  - Only as a last resort.
- dtrace!
  - Profiling showed a clear issue in the talloc destructor being fired on connection close.

Follow-up: Patch Submitted

- I submitted a patch that fixed the main issue causing the CPU DOS.

- It was quite easy to find, once I knew where to look.
  - I ended up debugging it without the talloc output about leaks.

- There was another related memory leak found by us; that was fixed also.
  - Libumem was used to find the actual leak, it was faster than valgrind.

Example Problem + Solution: Groups.

- Solaris 10 only allowed a user to be in 32 groups.
  - If setgroups got called with more it killed the process.
  - The Samba Team thought this was a security issue: People should be in the groups AD says.
  - The security issue wasn’t a concern for us.
  - So we patched our version and went about our life.

- Key point:
  - In an open source world we can “agree to disagree”.
  - Work together as you can.
  - Agree to disagree as you must.

Example Problem: Create + Notify

- SMB 2.002.
  - Compounded Create + Notify.
  - Every so often, our servers old weren’t replying correctly.
    - Causing disconnects
    - And builds to fall over. (Those 24hr ones.)
  - We actually diagnosed the issue. But no vendors took real notice.
  - What can we do?
    - Even by the time we GOT a fix from a vendor, it only fixed the bug sometimes.
  - We really wanted to rollout SMB2 badly, to speedup our builds.
    - Judged a “Critical Priority” within the company.

Solution: Create + Notify

- Eventually, we decided to just do it ourselves.
  - We worked in house to develop an awful prototype patch for Samba that showed the issue.
  - With the help of a few Samba Team members, we rewrote the patch into something that is usable.
  - Then we worked with the entire Samba Team for final QA.
  - The whole process took about 2 weeks, if not less.

Conclusion

- Samba + OpenSolaris has made our storage infrastructure much more effective.

- It has been a great way to give back to the community.

- Working with the Samba Team has been a joy.

- The powers that OpenSolaris and our team bring add a unique capability to the Samba Team.

Questions?

?

Thank you for attending!

!