Bestman 2.3.0 Scalability Testing

These are the various updates from Jeff Dost on the scalability tests he performed against Bestman 2.3.0

Update 1
==========

Let me describe a bit what I've done so far and let me know if my approach made sense and where we should go from here. Right now I tried to simulate ~50 concurrent requests to the test bestman server. I did this by creating a bash script that can call n lcg-ls commands from a particular node roughly the same time by starting them all as background processes.

Doing a single lcg-ls command takes about a tenth of a second (bestman is pointing to a dir in ramdisk). Doing ~50 lcg-ls commands at once from the same client node took about 2 seconds for them all to complete. This did not produce any errors for either release or developer versions of bestman, and there were no noticeable load issues.

I then moved the test to run from multiple client nodes at roughly the same time by calling the multi-ls script in a cron job. Running the script with n=7 concurrent lcg-ls commands per node over 8 nodes total gave 56 ~concurrent lcg-ls requests.

Again, this produced no errors, all were successful. However I only tried one "distributed" trial so far.

Since the original instructions were sparse I have a few concerns.

1. Does the bestman server or client do any kind of smart caching I should know about that could make these results deceiving?
2. if a single lcg-ls takes a fraction of a second, how can I really be sure I'm getting 50 concurrent requests at the server as apposed to them just being sequential
3. client and test server nodes used were all on the UCSD T2 network. Should I think about moving these tests to the WAN? I imagine it would be much harder to get simultaneous requests to happen in that case.

Update 2
==========

Here is a summary of the comparison between bestman2-server-2.2.1-4 (rel) and bestman2-server-2.3.0-6 (dev).

I ran the tests using the same setup I described previously. This time I started with 70 concurrent client lcg_ls commands and for each iteration increased the concurrency by a factor of 10 until I reached 700 concurrent lcg_ls.

Doing this reveals that 2.3 appears less stable than 2.2. Attached are a few plots. One compares the max %User CPU performance on the server, one is total server time per trial to serve all the requests, and one is the max time an lcg_ls command took on the client side time per trial.

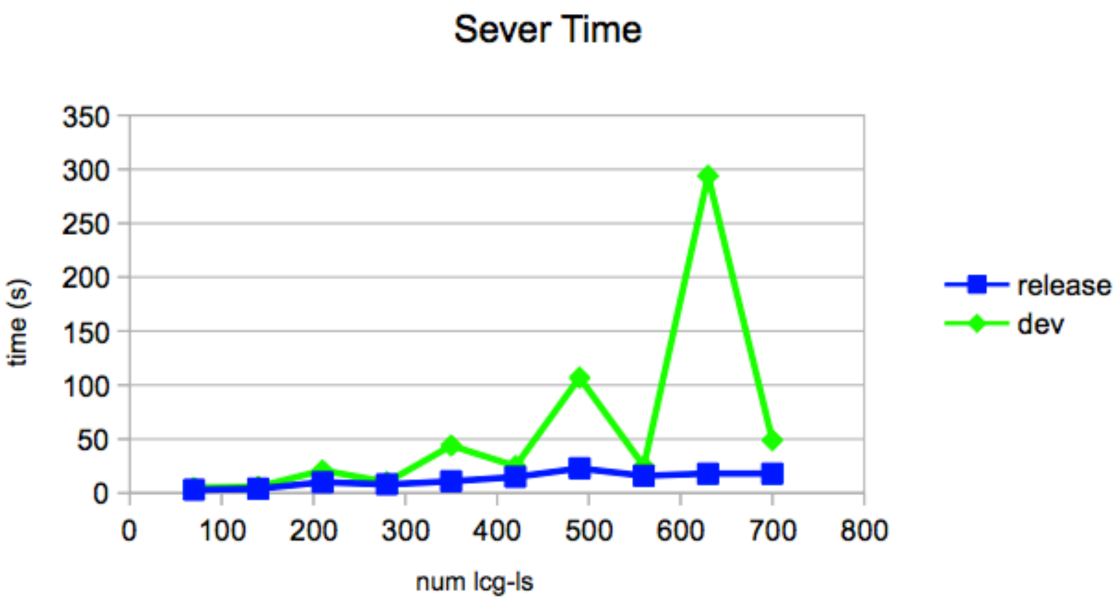
As you can see, as concurrency increases, 2.2 remains stable, but 2.3 varies wildly. The drop in max CPU may be misleading here since it was lower because really it was spread out much longer over time.

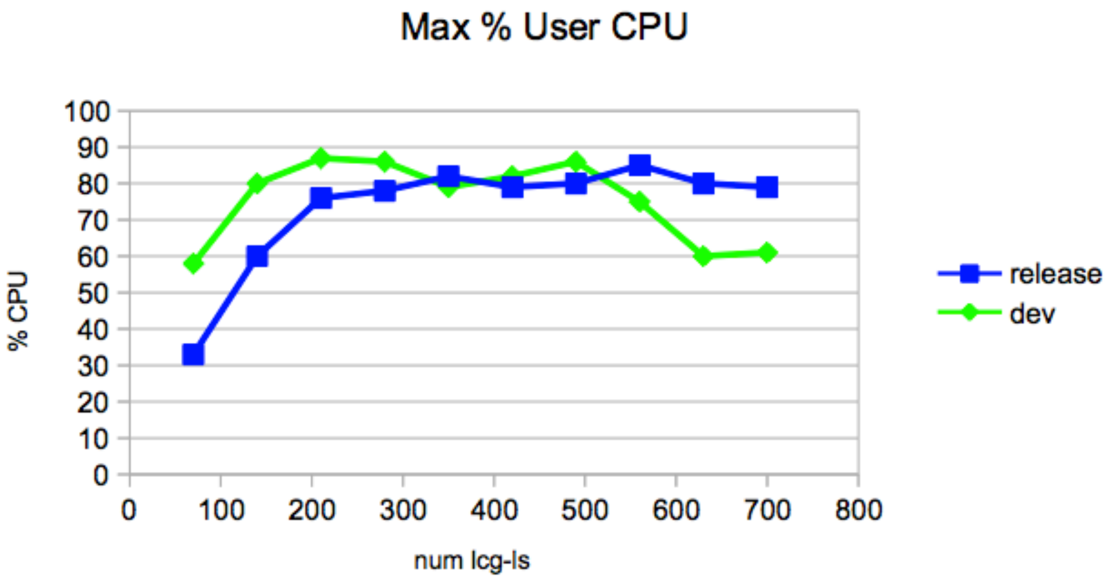
One other major observation, 2.2 was able to serve every request on every trial successfully. For 2.3 on the last trial about half of the lcg_ls commands failed. Here are the number of failures observed from the clients followed by each error message:
269 [SE][Ls][] httpg://cabinet-10-10-3.t2.ucsd.edu:8443/srm/v2/server: CGSI-gSOAP: Could not open connection !

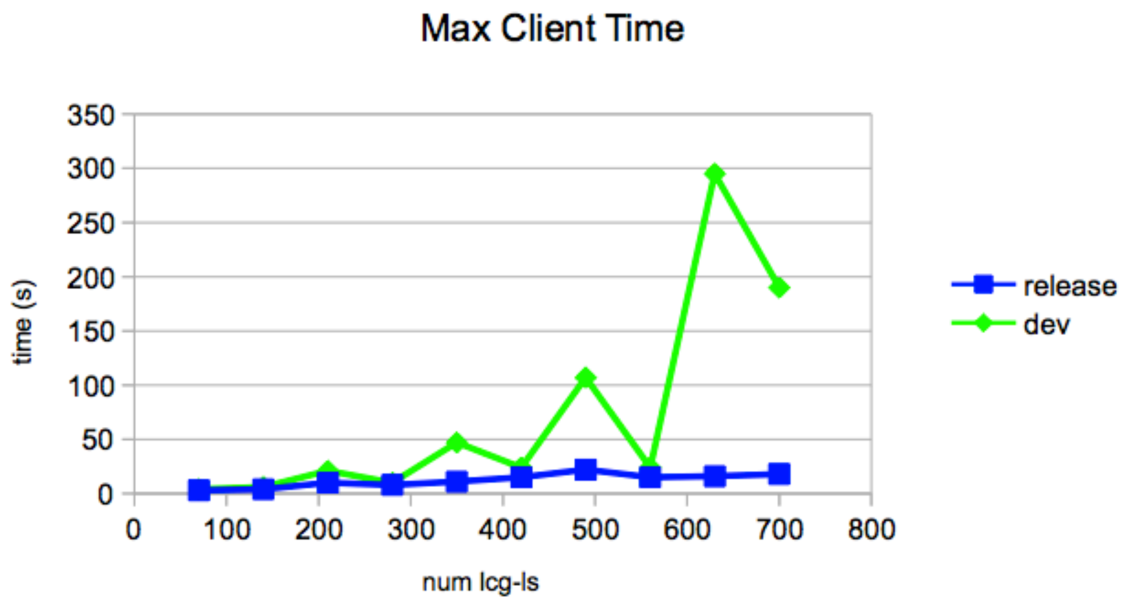
64 [SE][Ls][] httpg://cabinet-10-10-3.t2.ucsd.edu:8443/srm/v2/server: CGSI-gSOAP running on uaf-3.t2.ucsd.edu reports could not open connection to cabinet-10-10-3.t2.ucsd.edu

3 httpg://cabinet-10-10-3.t2.ucsd.edu:8443/srm/v2/server: not mapped./DC=org/DC=doegrids/OU=People/CN=Jeffrey M. Dost 948199
*** glibc detected *** double free or corruption (fasttop): 0x089e1008 ***

21 [SE][Ls][SRM_AUTHORIZATION_FAILURE] httpg://cabinet-10-10-3.t2.ucsd.edu:8443/srm/v2/server: not mapped./DC=org/DC=doegrids/OU=People/CN=Jeffrey M. Dost 948199


Sever Time
time (s)
num lcg-ls
release
dev
0
50
100
150
200
250
300
350
0
100
200
300
400
500
600
700
800


Max % User CPU
% CPU
num lcg-ls
release
dev
0
10
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600
700
800



If you'd like any more details, please let me know. I eyeballed the bestman server logs to see if there were any more detailed error messages but didn't see anything obvious on that side.

They were both pointing to our production GUMS server at UCSD.
ulimit -n
1024

Update 3
========
I ran a few more tests, watching the number of open fd's on the server with lsof. I re-ran 7 more trials of 700 concurrent lcg-ls client commands. This time the performance was much more comparable to the release version on average. There was only one trial that took over 5 minutes for all client commands to complete instead of ~20 seconds total, and for that run, concurrent open fd's got up to about 900. Interestingly I saw no errors this time, it just took a while for some of the lcg-ls commands to complete.

This definitely seemed to be approaching my default fd limits (1024) so it very possibly could have been the cause of my poor performance before.

I ran 7 more trials with bestman user fd ulimit increased to 4096. This time there were no problems for any of the trial runs.

One other general observation, the number of concurrent fd's in general averaged between 400-500. It was only that one anomalous run where bestman seemed to need 2x more fd's. I don't have any explanation for why the number seems to vary so much, but as I mentioned in the previous emails, my setup isn't truly performing 700 concurrent client commands occurring at exactly the same time, so perhaps they just happened to be more "simultaneous" for that run.