Managing a BOINC Server

Experiences at World Community Grid

September 10, 2008

world community grid™

BOINC

Introduction to World Community Grid

- World Community Grid – funded by IBM's Corporate Citizenship & Corporate Affairs
- Research projects who want to tap the power available from volunteers but do not have the expertise or desire to run their own project can apply to run on World Community Grid at: http://www.worldcommunitygrid.org/projects_showcase/viewSubmitAProposal.do
- Kevin Reed – Assigned to World Community Grid since before it launched in November 2004
- Over 235,000 results sent/received daily (~200 years of cpu time returned daily)
- Now Running at just under 200 TFlops

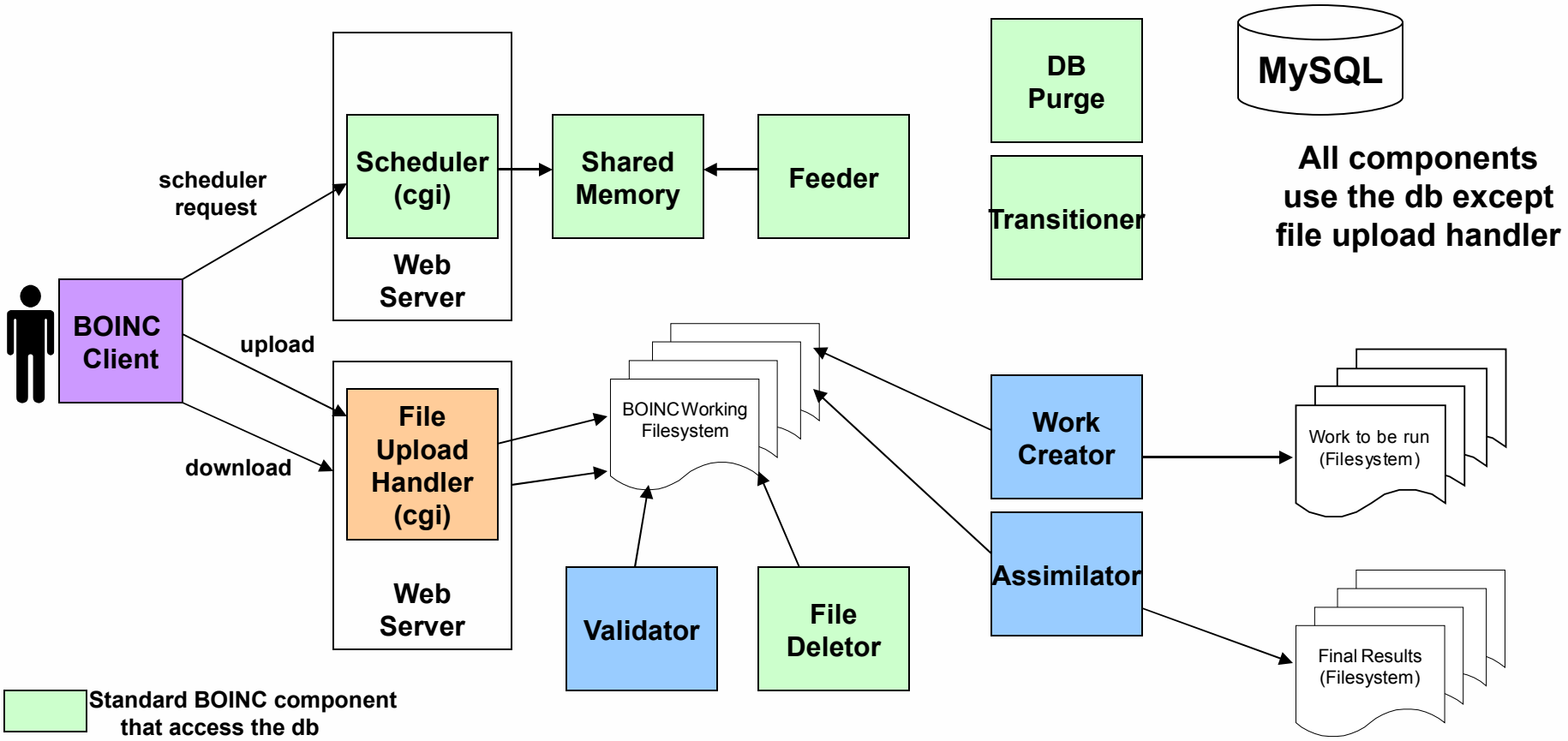
Running a volunteer BOINC project

- Purpose - Get lots of work processed through the grid quickly
- Things to consider
  - Volunteered computers are heterogeneous
    - Different OS, OS versions and processors
    - Varying amounts of Ram
    - Varying availability
  - What resources do you have to run your project
    - Server processing, storage and memory capacity
  - Members need to feel involved
    - Need to believe that you are using their computer in a valuable way
    - Need to believe that they are being fairly acknowledge for this contribution

Workunit Lifecycle

1. Load a workunit
2. Create replicas of the workunit
3. Buffer replicas to send
4. Client requests work (assign replicas to client)
5. Download to client
6. Process on client
7. Upload result from client
8. Report result upload
9. Determine if there is a quorum of results
10. Compare results and award credit to members
11. Generate new replicas if consensus not reached
12. Assimilate result if consensus is reached
13. Delete workunit and result data
14. Purge records from the database

Major Components



sched

world community grid

BOINC

Load a workunit – Create Work

- Major Fields
  - Input Files (Workunit)
  - Output Files (Result)
  - Estimated Flops (Flops)
    - Can be hard to provide accurately – but it is critical!
    - *We were not good at this – so we use as our estimate the computed flops of recently returned results*
    - Client 'duration correct factor'
    - *We have found that workunits that average about 5 hours of cpu time to be well received by members*
  - Resource Limits (Disk, Ram, Flops)
    - Returned as error if any of these are exceeded

Load a workunit – Create Work

- Major Fields - continued
  - Batch Id
    - Used to delay file deletion until a group of workunits is done (useful if they share some of the same download files)
  - Deadline – a tricky balancing act
    - Some systems are slow at returning results
    - Researchers want the answer quickly
    - Users want credit quickly
    - *We have deadlines that are 12 days long for workunits that average 8 hours of cpu time*
    - *We have found that about 75% of our results are returned within 3 days*

Load a workunit – Create Work

- Workunit Scheduling ‘A bad case’
  - 2 replicas created
  - 1 returned quickly and successfully
  - 1 is never returned so a new copy is sent
  - That copy is reported as an error after half the deadline goes by
  - Another copy is generated but when validation occurs the results don’t match so a third copy is sent
  - That copy finally matches the first and the workunit completes
  - However, this could be 3-4 times the deadline set for the work
- When you are sending 100,000’s of workunits – this does happen!!!
- Use the ‘reliable’ mechanisms discussed below to minimize this

Load a workunit – Create Work

- Major Fields - continued
  - Min Quorum (more info in discussion about validator)
  - Num replicas
    - allowed to be >= min quorum but I strongly advise setting it equal to min quorum
  - Priority
- *Our create work script monitors the rate that workunits are being distributed and four times a day it will load in enough work for each project so that there is roughly a 24 hours supply of work ready to send*
- BOINC Website References
  - http://boinc.berkeley.edu/trac/wiki/WorkGeneration
  - http://boinc.berkeley.edu/trac/wiki/JobIn

Create replicas of the workunit – Transitioner

- The transitioner acts on the workunit and is the primary daemon that advances the state of a workunit

- In this step of the process, the transitioner will see that the workunit has no replicas created for the workunit. It will create ‘Num replicas’ copies.

Buffer replicas to send – Feeder and Shared Memory

- The shared memory segment is used to store data that is needed by the scheduler but requires expensive database queries to obtain
- The feeder is a deamon whose purpose is to keep the shared memory populated with data for quick access by the scheduler

Buffer replicas to send – Feeder and Shared Memory

- Command line
  - feeder -allapps -priority_order_create_time -mod 2 0 -d 3
  - -allapps = ensures that each application will be present in the shared memory segment with the number of ‘slots’ in proportion to the ‘weight’ field for the application on the app table
  - -priority_order_create_time = when querying the database for results to send sort by result.priority desc and workunit.create_time asc
  - -mod 2 0 = We have two web servers and this is server 0 (our other server would have –mod 2 1)
  - Additional options at: http://boinc.berkeley.edu/trac/wiki/BackendPrograms

Buffer replicas to send – Feeder and Shared Memory

- Important Settings – configuration file
  - <shmem_work_items>N</shmem_work_items>
  - <feeder_query_size>N</feeder_query_size>
  - The appropriate settings to use are relative to the rate you send results and how many applications you have work ready to send
  - *We have 5 active research applications and send about 240,000 results/day (or 2.8 per second)*
  - *We use: shmem_work_items=1800, feeder_query_size=2*shmem_work_items*
  - More information at: http://boinc.berkeley.edu/trac/wiki/ProjectConfigFile

Client requests work (assign replicas to client) - scheduler

- Clients periodically request work
- The server scheduler takes into account various parameters to assign work to the client requesting work
- User Preferences
  - Does the user want work from this application (for projects running multiple applications)
- Client Capabilities
  - Available Memory (actual ram adjusted by applicable user preferences)
  - Available Disk Space (actual disk space adjust by applicable user preferences)
  - Is their an application binary available to send to this computer (co-processors, operating system, processors, etc)

Client requests work (assign replicas to client) - scheduler

- Homogenous Class
  - Set at the application level, evaluated for each client when it makes a request
  - BOINC provides 3 homogenous classes:
    - ✓ 0: not scored (some applications do not need this)
    - ✓ 1: fine grained (os and processor versions – 80 different outcomes)
    - ✓ 2: coarse grained (windows, linux, mac-ppc, mac-intel)
    - ✓ Projects can extend if desired.
  - Client is evaluated based upon operating system, processor type and additional factors if the project decides to extend
  - More info at http://boinc.berkeley.edu/trac/wiki/HomogeneousRedundancy

Client requests work (assign replicas to client) - scheduler

- Reliable Computers
  - Does the client has a recent history of returning valid results quickly?
    - ✓ <reliable_max_avg_turnaround>*75600*</reliable_max_avg_turnaround>
      - Computers that return results within 21 hours of assignment
    - ✓ <reliable_max_error_rate>*0.001*</reliable_max_error_rate>
      - Recent history of over 99.9% valid results returned
    - ✓ A good choice for these values are ones that allow about 15% of your project's recent credit to come from 'reliable' computers
  - Is there a result that needs a reliable computers?
    - ✓ <reliable_on_priority>*10*</reliable_on_priority>
      - Results with a priority equal or greater to this value will only be sent to 'reliable' computers
  - Reduce the deadline when assigned to a reliable computers
    - ✓ <reliable_reduced_delay_bound>*0.20*</reliable_reduced_delay_bound>
      - When assigned with this setting, reliable results will have a deadline of 20% of the original deadline – so 12 days becomes 2.4 days
  - The scheduler will give a preference to assign a reliable computer a result that needs a reliable computer
  - More Info at: http://boinc.berkeley.edu/trac/wiki/ProjectOptions

Client requests work (assign replicas to client) - scheduler

- Additional settings
  - <daily_result_quota>*80*</daily_result_quota>
    - ✓ This is the max number of results per processor that can be sent to a client each calendar day. This should be set to the lowest value that will allow the most powerful computer attached to your project to get enough work to run at 100% for 24 hours. Useful for dealing limiting the impact of cyclers.
  - <resend_lost_results>*1*</resend_lost_results>
    - ✓ For various reasons, a result can be marked in the database as assigned to a computer, but the computer doesn't know about it (interrupted communication, re-installation of the client, re-imaged computer, etc). This setting will cause the server to notify the computer about these results
    - ✓ This causes a moderately heavy db query so it is optional, however, the biggest delay in finishing a workunit are results that 'time-out'. This function reduces the number of those considerable.
    - ✓ *We use this setting. I recommend that you use it unless your database is under very heavy load*

Client requests work (assign replicas to client) - scheduler

- Additional settings - continued
  - <next_rpc_delay>*345600*</next_rpc_delay> and
    <send_result_abort>*1*</send_result_abort>
    - ✓ There are times when a project makes a mistake and has to cancel workunits or a user shuts down their computer when they go on vacation. In these cases, the work in progress can become unneeded or in the event of a project mistake, useless. Next_rpc_delay ensures that any client with boinc running and a internet connection available will talk to your project again after at most X seconds. Send result abort instructs the project to send messages to the client telling it to abort work under various conditions if the work will no longer be useful.

Client requests work (assign replicas to client) - scheduler

- Additional settings - continued
  - <one_result_per_user_per_wu/> or <one_result_per_host_per_wu/>
    - ✓ Computers owned by a user (or the same computer) are more likely to experience similar errors, these settings ensure that results from one workunit are sent to different users or computers
    - ✓ *We use <one_result_per_user_per_wu/>*
  - <max_wus_in_progress> N </max_wus_in_progress>
    - ✓ Optional – allows projects to limit the max number of results a client can cache (per processor). Use of this will annoy some users.
  - <min_sendwork_interval> and <max_wus_to_send>
    - ✓ Controls how often a client can request work and for each request, how many replicas can be assigned to the client. This is useful to limiting the impact of 'cyclers' but too restrictive settings will annoy people with powerful computers
    - ✓ *We use 60 seconds for min_sendwork_interval and 15 replicas per request*
- More information is available: http://boinc.berkeley.edu/trac/wiki/ProjectOptions

Download to client – Apache HTTP Server

- Once assigned work, the client will identify files for the replica that it doesn't currently have locally. It will download those that it doesn't have
- Files can be marked as sticky for the workunit. These are never deleted by the client
- Dynamic compression of file downloads
  - Your apache http server can be instructed to perform compression on the fly using the mod_deflate module
  - This saves user and project bandwidth
  - Load on server is surprisingly moderate
  - Our settings:

```
DeflateMemLevel 2
DeflateCompressionLevel 2
<Directory /boinc/data/download>
    SetOutputFilter DEFLATE
    SetEnvIfNoCase Request_URI \.(?:gz|gif|jpg|jpeg|png|jp2)$ no-gzip dont-vary
</Directory>
```

- More information at http://boinc.berkeley.edu/trac/wiki/FileCompression

Process on client - BOINC client

- The work runs (See Rom's lecture for information)

Upload result from client – file_upload_handler

- The client uploads the result to the file_upload_handler
- Note that the file_upload_handler does not require db access
- If the result template is marked with <gzip_when_done/> then file will be compressed before upload
- See http://boinc.berkeley.edu/trac/wiki/XmlFormat#Files for info

Upload result from client – file_upload_handler

- NOTE – the <gzip_when_done/> option was added with BOINC 5.8 so you may get some results not compressed. You can use the code similar to the following to open the file whether it is compressed or not:

```
int read_wcg_file_string(string file, string* result) {
    FILE *infile;
    char cmd[512] = '\0';
    char buf[4096];
    result->erase();

    //build the command
    strcat(cmd, "gzip -dcf ");
    strcat(cmd, file.c_str());

    infile = popen(cmd, "r");
    if (infile == NULL) return ERR_FOPEN;

    while( fgets(buf, 4096, infile) != NULL ) result->append(buf);

    result->append("\0");
    if (pclose(infile) != 0) return 1;
    return 0;
}
```

Report result upload – scheduler

- The client makes a scheduler request to notify the server that the result is done and has been uploaded (or report if the result ended with an error)
- Note that this is a second step that occurs sometime after the result files have been successfully uploaded
- This update causes the transitioner to be run for the workunit

Determine if there is a quorum of results - transitioner

- As each result is reported returned, the transitioner runs.
- If the result was a success and if it sees that there are at least ‘min_quorum’ results successfully returned, then it will trigger the validator to run for the workunit
- If the result was an error and that the sum: # results ready to send + # results in progress + # results returned successfully (validation pending or inconclusive) < Num Replicas, then it will generate enough replicas to make that sum equal to Num Replicas again

Determine if there is a quorum of results - transitioner

- Set <reliable_priority_on_over>*10*</reliable_priority_on_over> in order to have the priority for these additional replicas set to: X + Workunit.priority
  - Use this to cause the additional replicas to be sent to reliable clients
- This setting in conjuction with the other ‘reliable’ settings for the scheduler are key to significantly reducing the number of ‘bad case’ scenarios for a workunit

Compare results and award credit to members - validator

- The validator is responsible for determining if the results returned for the workunit reach a consensus
- If min_quorum >= 2 then you can compare between the different results
  - Fuzzy validation = if you know that the results can vary by a certain amount due to running on different platforms, then you can compare the ranges of values
  - Bitwise validation = if you a computation that diverges quickly for small values, then you will need to set the appropriate homogenous redundancy settings and make exact comparisons between files
- Note: Credit awarded is based off an agreement between the claimed credits between the two results

Compare results and award credit to members - validator

- UPCOMING: If min_quorum = 1, then you should implement some of the following in the check_one method
  - A set of data run periodically throughout the workunit that can be used to quickly determine on the server if the computation is proceeding correctly (think floating point errors due to over clocking)
  - A method to determine if the result was calculated based upon the input files sent (think of a user trying to get lots of credit by skipping the actual computation and always returning a result that passed the above check)
- This mechanism will automatically send out a second copy of the workunit if the computer the original copy was sent to does not have a sufficiently high rate of recent valid results
- Also includes a random sampling during validation that will send a second copy periodical to check the result

Generate new replicas if consensus not reached

- If no consensus is reached for the results returned, then increase the value of ‘Num Replicas’ by one and have the transitioner generate an additional result to send
- The other results are now marked ‘inconclusive’ and will be re-examined once the additional result is returned

Assimilate result if consensus is reached - assimilator

- If consensus is reached, then one result is identified as the ‘canonical’ result that represents the correct computation
- The workunit is flagged for assimilation
- The assimilator will then run. The assimilator exists to allow a project to archive the result in any manner that the project needs.

Delete workunit and result data – file_deletor

- After assimilation, the transitioner will determine that the workunit is done and flag the workunit and results for deletion
- All workunits with a input file marked as “no_delete” will not be deleted
- 1 hours after file_deletor is started and every 24 hours there after, the file_deletor will identify the time the oldest workunit in the db was created. It will then execute a find on the upload directory and delete all results older than this time

Delete workunit and result data – file_deletor

- If file_deletor uses the option -dont_delete_batches, then workunits will not be deleted unless their batch id = 0 (you need to have an external process make the change)

- You can delay the deletion of files for a period of time after the last result is returned using the <delete_delay_hours>. This is useful in case you have some error occur and you need to correct
  - *Inevitably At some point you will have a problem*
  - *We use 24 hours*

Purge records from the database – db_purge

- Once the files have been deleted for the workunit and results, then the records are eligible to be deleted
- They will be deleted based on the value of the command line setting: -min_age_days n
- Members like to see a bit of their history so keep the records around for awhile – however make sure that you do delete the records otherwise your database will get very big

Other things good to know

- Beta Testing (users can opt-in to participating in beta testing). This is a parameter set on the app table
- DBVisualizer is a great tool to query the database with (free): http://www.minq.se/products/dbvis/download/
- Stop/Start scripts: http://boinc.berkeley.edu/trac/wiki/StartTool

Questions?