GridMix

Table of contents

1 Overview.......... 2
2 Usage.......... 2
3 General Configuration Parameters.......... 3
4 Job Types.......... 5
5 Job Submission Policies.......... 6
6 Emulating Users and Queues.......... 8
7 Emulating Distributed Cache Load.......... 10
8 Configuration of Simulated Jobs.......... 10
9 Emulating Compression/Decompression.......... 10
10 Emulating High-Ram jobs.......... 11
11 Emulating resource usages.......... 12
12 Simplifying Assumptions.......... 13
13 Appendix.......... 14

1 Overview

GridMix is a benchmark for Hadoop clusters. It submits a mix of synthetic jobs, modeling a profile mined from production loads.

There exist three versions of the GridMix tool. This document discusses the third (checked into `src/contrib`), distinct from the two checked into the `src/benchmarks` sub-directory. While the first two versions of the tool included stripped-down versions of common jobs, both were principally saturation tools for stressing the framework at scale. In support of a broader range of deployments and finer-tuned job mixes, this version of the tool will attempt to model the resource profiles of production jobs to identify bottlenecks, guide development, and serve as a replacement for the existing GridMix benchmarks.

To run GridMix, you need a MapReduce job trace describing the job mix for a given cluster. Such traces are typically generated by Rumen (see Rumen documentation). GridMix also requires input data from which the synthetic jobs will be reading bytes. The input data need not be in any particular format, as the synthetic jobs are currently binary readers. If you are running on a new cluster, an optional step generating input data may precede the run.

In order to emulate the load of production jobs from a given cluster on the same or another cluster, follow these steps:

1. Locate the job history files on the production cluster. This location is specified by the `mapred.job.tracker.history.completed.location` configuration property of the cluster.
2. Run Rumen to build a job trace in JSON format for all or select jobs.
3. Use GridMix with the job trace on the benchmark cluster.

Jobs submitted by GridMix have names of the form "`GRIDMIXnnnnnn`", where "`nnnnnn`" is a sequence number padded with leading zeroes.

2 Usage

Basic command-line usage without configuration parameters:

```
org.apache.hadoop.mapred.gridmix.Gridmix [-generate <size>] [-users <users-list>] <iopath>
 <trace>
```

Basic command-line usage with configuration parameters:

```
org.apache.hadoop.mapred.gridmix.Gridmix \
  -Dgridmix.client.submit.threads=10 -Dgridmix.output.directory=foo \
  [-generate <size>] [-users <users-list>] <iopath> <trace>
```

Note:

Configuration parameters like `-Dgridmix.client.submit.threads=10` and `-Dgridmix.output.directory=foo` as given above should be used *before* other GridMix parameters.

The `<iopath>` parameter is the working directory for GridMix. Note that this can either be on the local file-system or on HDFS, but it is highly recommended that it be the same as that for the original job mix so that GridMix puts the same load on the local file-system and HDFS respectively.

The `-generate` option is used to generate input data and Distributed Cache files for the synthetic jobs. It accepts standard units of size suffixes, e.g. `100g` will generate $100 * 2^{30}$ bytes as input data. `<iopath>/input` is the destination directory for generated input data and/or the directory from which input data will be read. HDFS-based Distributed Cache files are generated under the distributed cache directory `<iopath>/distributedCache`. If some of the needed Distributed Cache files are already existing in the distributed cache directory, then only the remaining non-existing Distributed Cache files are generated when `-generate` option is specified.

The `-users` option is used to point to a users-list file (see Emulating Users and Queues).

The `<trace>` parameter is a path to a job trace generated by Rumen. This trace can be compressed (it must be readable using one of the compression codecs supported by the cluster) or uncompressed. Use "-" as the value of this parameter if you want to pass an *uncompressed* trace via the standard input-stream of GridMix.

The class `org.apache.hadoop.mapred.gridmix.Gridmix` can be found in the JAR `contrib/gridmix/hadoop-gridmix-$VERSION.jar` inside your Hadoop installation, where `$VERSION` corresponds to the version of Hadoop installed. A simple way of ensuring that this class and all its dependencies are loaded correctly is to use the `hadoop` wrapper script in Hadoop:

```
hadoop jar <gridmix-jar> org.apache.hadoop.mapred.gridmix.Gridmix \
  [-generate <size>] [-users <users-list>] <iopath> <trace>
```

The supported configuration parameters are explained in the following sections.

3 General Configuration Parameters

Parameter	Description
`gridmix.output.directory`	The directory into which output will be written. If specified, `iopath` will be relative to this parameter. The submitting user must have read/write access to this directory. The user should also be mindful of any quota issues that may arise during a run. The default is "`gridmix`".
`gridmix.client.submit.threads`	The number of threads submitting jobs to the cluster. This also controls how many splits will be loaded into memory at a given time, pending the submit time in the trace. Splits are pre-generated to hit submission deadlines, so particularly dense traces may want more submitting threads. However, storing splits in memory is reasonably expensive, so you should raise this cautiously. The default is 1 for the SERIAL job-submission policy (see Job Submission Policies) and one more than the number of processors on the client machine for the other policies.
`gridmix.submit.multiplier`	The multiplier to accelerate or decelerate the submission of jobs. The time separating two jobs is multiplied by this factor. The default value is 1.0. This is a crude mechanism to size a job trace to a cluster.
`gridmix.client.pending.queue.depth`	The depth of the queue of job descriptions awaiting split generation. The jobs read from the trace occupy a queue of this depth before being processed by the submission threads. It is unusual to configure this. The default is 5.
`gridmix.gen.blocksize`	The block-size of generated data. The default value is 256 MiB.
`gridmix.gen.bytes.per.file`	The maximum bytes written per file. The default value is 1 GiB.
`gridmix.min.file.size`	The minimum size of the input files. The default limit is 128 MiB. Tweak this parameter if you see an error-message like "Found no satisfactory file" while testing GridMix with a relatively-small input data-set.
`gridmix.max.total.scan`	The maximum size of the input files. The default limit is 100 TiB.

Parameter	Description
`gridmix.task.jvm-options.enable`	Enables Gridmix to configure the simulated task's max heap options using the values obtained from the original task (i.e via trace).

4 Job Types

GridMix takes as input a job trace, essentially a stream of JSON-encoded job descriptions. For each job description, the submission client obtains the original job submission time and for each task in that job, the byte and record counts read and written. Given this data, it constructs a synthetic job with the same byte and record patterns as recorded in the trace. It constructs jobs of two types:

Job Type	Description
`LOADJOB`	A synthetic job that emulates the workload mentioned in Rumen trace. In the current version we are supporting I/O. It reproduces the I/O workload on the benchmark cluster. It does so by embedding the detailed I/O information for every map and reduce task, such as the number of bytes and records read and written, into each job's input splits. The map tasks further relay the I/O patterns of reduce tasks through the intermediate map output data.
`SLEEPJOB`	A synthetic job where each task does *nothing* but sleep for a certain duration as observed in the production trace. The scalability of the Job Tracker is often limited by how many heartbeats it can handle every second. (Heartbeats are periodic messages sent from Task Trackers to update their status and grab new tasks from the Job Tracker.) Since a benchmark cluster is typically a fraction in size of a production cluster, the heartbeat traffic generated by the slave nodes is well below the level of the production cluster. One possible solution is to run multiple Task Trackers on each slave node. This leads to the obvious problem that the I/O workload generated by the synthetic jobs would thrash the slave nodes. Hence the need for such a job.

The following configuration parameters affect the job type:

Parameter	Description
`gridmix.job.type`	The value for this key can be one of LOADJOB or SLEEPJOB. The default value is LOADJOB.
`gridmix.key.fraction`	For a LOADJOB type of job, the fraction of a record used for the data for the key. The default value is 0.1.
`gridmix.sleep.maptask-only`	For a SLEEPJOB type of job, whether to ignore the reduce tasks for the job. The default is `false`.
`gridmix.sleep.fake-locations`	For a SLEEPJOB type of job, the number of fake locations for map tasks for the job. The default is 0.
`gridmix.sleep.max-map-time`	For a SLEEPJOB type of job, the maximum runtime for map tasks for the job in milliseconds. The default is unlimited.
`gridmix.sleep.max-reduce-time`	For a SLEEPJOB type of job, the maximum runtime for reduce tasks for the job in milliseconds. The default is unlimited.

5 Job Submission Policies

GridMix controls the rate of job submission. This control can be based on the trace information or can be based on statistics it gathers from the Job Tracker. Based on the submission policies users define, GridMix uses the respective algorithm to control the job submission. There are currently three types of policies:

Job Submission Policy	Description
`STRESS`	Keep submitting jobs so that the cluster remains under stress. In this mode we control the rate of job submission by monitoring the real-time load of the cluster so that we can maintain a stable stress level of workload on the cluster. Based on the statistics we gather we define if a cluster is *underloaded* or *overloaded*. We consider a cluster *underloaded* if and only if the following three conditions are true: 1. the number of pending and running jobs are under a threshold TJ 2. the number of pending and running maps are under threshold TM 3. the number of pending and running reduces are under threshold TR The thresholds TJ, TM and TR are proportional to the size of the cluster and map, reduce slots capacities respectively. In case of a cluster being

Job Submission Policy	Description
	overloaded, we throttle the job submission. In the actual calculation we also weigh each running task with its remaining work - namely, a 90% complete task is only counted as 0.1 in calculation. Finally, to avoid a very large job blocking other jobs, we limit the number of pending/waiting tasks each job can contribute.
`REPLAY`	In this mode we replay the job traces faithfully. This mode exactly follows the time-intervals given in the actual job trace.
`SERIAL`	In this mode we submit the next job only once the job submitted earlier is completed.

The following configuration parameters affect the job submission policy:

Parameter	Description
`gridmix.job-submission.policy`	The value for this key would be one of the three: STRESS, REPLAY or SERIAL. In most of the cases the value of key would be STRESS or REPLAY. The default value is STRESS.
`gridmix.throttle.jobs-to-tracker-ratio`	In STRESS mode, the minimum ratio of running jobs to Task Trackers in a cluster for the cluster to be considered *overloaded*. This is the threshold TJ referred to earlier. The default is 1.0.
`gridmix.throttle.maps.task-to-slot-ratio`	In STRESS mode, the minimum ratio of pending and running map tasks (i.e. incomplete map tasks) to the number of map slots for a cluster for the cluster to be considered *overloaded*. This is the threshold TM referred to earlier. Running map tasks are counted partially. For example, a 40% complete map task is counted as 0.6 map tasks. The default is 2.0.
`gridmix.throttle.reduces.task-to-slot-ratio`	In STRESS mode, the minimum ratio of pending and running reduce tasks (i.e. incomplete reduce tasks) to the number of reduce slots for a cluster for the cluster to be considered *overloaded*. This is the threshold TR referred to earlier. Running reduce tasks are counted partially. For example, a 30% complete reduce task is counted as 0.7 reduce tasks. The default is 2.5.
`gridmix.throttle.maps.max-slot-share-per-job`	In STRESS mode, the maximum share of a cluster's map-slots capacity that can be counted toward a job's

Parameter	Description
	incomplete map tasks in overload calculation. The default is 0.1.
`gridmix.throttle.reducess.max-slot-share-per-job`	In STRESS mode, the maximum share of a cluster's reduce-slots capacity that can be counted toward a job's incomplete reduce tasks in overload calculation. The default is 0.1.

6 Emulating Users and Queues

Typical production clusters are often shared with different users and the cluster capacity is divided among different departments through job queues. Ensuring fairness among jobs from all users, honoring queue capacity allocation policies and avoiding an ill-behaving job from taking over the cluster adds significant complexity in Hadoop software. To be able to sufficiently test and discover bugs in these areas, GridMix must emulate the contentions of jobs from different users and/or submitted to different queues.

Emulating multiple queues is easy - we simply set up the benchmark cluster with the same queue configuration as the production cluster and we configure synthetic jobs so that they get submitted to the same queue as recorded in the trace. However, not all users shown in the trace have accounts on the benchmark cluster. Instead, we set up a number of testing user accounts and associate each unique user in the trace to testing users in a round-robin fashion.

The following configuration parameters affect the emulation of users and queues:

Parameter	Description
`gridmix.job-submission.use-queue-in-trace`	When set to `true` it uses exactly the same set of queues as those mentioned in the trace. The default value is `false`.
`gridmix.job-submission.default-queue`	Specifies the default queue to which all the jobs would be submitted. If this parameter is not specified, GridMix uses the default queue defined for the submitting user on the cluster.
`gridmix.user.resolve.class`	Specifies which `UserResolver` implementation to use. We currently have three implementations: 1. `org.apache.hadoop.mapred.gridmix.EchoUserResol` - submits a job as the user who submitted the original job. All the users of the production cluster identified in the job trace must also have accounts on the benchmark cluster in this case. 2. `org.apache.hadoop.mapred.gridmix.SubmitterUser` - submits all the jobs as current GridMix user. In

Parameter	Description
	this case we simply map all the users in the trace to the current GridMix user and submit the job. 3. `org.apache.hadoop.mapred.gridmix.RoundRobinUse` - maps trace users to test users in a round-robin fashion. In this case we set up a number of testing user accounts and associate each unique user in the trace to testing users in a round-robin fashion. The default is `org.apache.hadoop.mapred.gridmix.SubmitterUserRes`

If the parameter `gridmix.user.resolve.class` is set to `org.apache.hadoop.mapred.gridmix.RoundRobinUserResolver`, we need to define a users-list file with a list of test users. This is specified using the `-users` option to GridMix.

> **Note:**
>
> Specifying a users-list file using the `-users` option is mandatory when using the round-robin user-resolver. Other user-resolvers ignore this option.

A users-list file has one user per line, each line of the format:

```
<username>
```

For example:

```
user1
user2
user3
```

In the above example we have defined three users `user1`, `user2` and `user3`. Now we would associate each unique user in the trace to the above users defined in round-robin fashion. For example, if trace's users are `tuser1`, `tuser2`, `tuser3`, `tuser4` and `tuser5`, then the mappings would be:

```
tuser1 -> user1
tuser2 -> user2
tuser3 -> user3
tuser4 -> user1
tuser5 -> user2
```

For backward compatibility reasons, each line of users-list file can contain username followed by groupnames in the form username[,group]*. The groupnames will be ignored by Gridmix.

7 Emulating Distributed Cache Load

Gridmix emulates Distributed Cache load by default for LOADJOB type of jobs. This is done by precreating the needed Distributed Cache files for all the simulated jobs as part of a separate MapReduce job.

Emulation of Distributed Cache load in gridmix simulated jobs can be disabled by configuring the property `gridmix.distributed-cache-emulation.enable` to `false`. But generation of Distributed Cache data by gridmix is driven by `-generate` option and is independent of this configuration property.

Both generation of Distributed Cache files and emulation of Distributed Cache load are disabled if:

- input trace comes from the standard input-stream instead of file, or
- `<iopath>` specified is on local file-system, or
- any of the ascendant directories of the distributed cache directory i.e. `<iopath>/distributedCache` (including the distributed cache directory) doesn't have execute permission for others.

8 Configuration of Simulated Jobs

Gridmix3 sets some configuration properties in the simulated Jobs submitted by it so that they can be mapped back to the corresponding Job in the input Job trace. These configuration parameters include:

Parameter	Description
`gridmix.job.original-job-id`	The job id of the original cluster's job corresponding to this simulated job.
`gridmix.job.original-job-name`	The job name of the original cluster's job corresponding to this simulated job.

9 Emulating Compression/Decompression

MapReduce supports data compression and decompression. Input to a MapReduce job can be compressed. Similarly, output of Map and Reduce tasks can also be compressed. Compression/Decompression emulation in GridMix is important because emulating compression/decompression will effect the CPU and Memory usage of the task. A task emulating compression/decompression will affect other tasks and daemons running on the same node.

Compression emulation is enabled if `gridmix.compression-emulation.enable` is set to `true`. By default compression emulation is enabled for jobs of type *LOADJOB*. With compression emulation enabled, GridMix will now generate compressed text data with a constant compression ratio. Hence a simulated GridMix job will now emulate compression/decompression using compressible text data (having a constant compression ratio), irrespective of the compression ratio observed in the actual job.

A typical MapReduce Job deals with data compression/decompression in the following phases

- `Job input data decompression:` GridMix generates compressible input data when compression emulation is enabled. Based on the original job's configuration, a simulated GridMix job will use a decompressor to read the compressed input data. Currently, GridMix uses `mapreduce.input.fileinputformat.inputdir` to determine if the original job used compressed input data or not. If the original job's input files are uncompressed then the simulated job will read the compressed input file without using a decompressor.
- `Intermediate data compression and decompression:` If the original job has map output compression enabled then GridMix too will enable map output compression for the simulated job. Accordingly, the reducers will use a decompressor to read the map output data.
- `Job output data compression:` If the original job's output is compressed then GridMix too will enable job output compression for the simulated job.

The following configuration parameters affect compression emulation

Parameter	Description
gridmix.compression-emulation.enable	Enables compression emulation in simulated GridMix jobs. Default is true.

With compression emulation turned on, GridMix will generate compressed input data. Hence the total size of the input data will be lesser than the expected size. Set `gridmix.min.file.size` to a smaller value (roughly 10% of `gridmix.gen.bytes.per.file`) for enabling GridMix to correctly emulate compression.

10 Emulating High-Ram jobs

MapReduce allows users to define a job as a High-Ram job. Tasks from a High-Ram job can occupy multiple slots on the task-trackers. Task-tracker assigns fixed virtual memory for each slot. Tasks from High-Ram jobs can occupy multiple slots and thus can use up more virtual memory as compared to a default task.

Emulating this behavior is important because of the following reasons

- Impact on scheduler: Scheduling of tasks from High-Ram jobs impacts the scheduling behavior as it might result into slot reservation and slot/resource utilization.
- Impact on the node : Since High-Ram tasks occupy multiple slots, trackers do some bookkeeping for allocating extra resources for these tasks. Thus this becomes a precursor for memory emulation where tasks with high memory requirements needs to be considered as a High-Ram task.

High-Ram feature emulation can be disabled by setting `gridmix.highram-emulation.enable` to `false`.

11 Emulating resource usages

Usages of resources like CPU, physical memory, virtual memory, JVM heap etc are recorded by MapReduce using its task counters. This information is used by GridMix to emulate the resource usages in the simulated tasks. Emulating resource usages will help GridMix exert similar load on the test cluster as seen in the actual cluster.

MapReduce tasks use up resources during its entire lifetime. GridMix also tries to mimic this behavior by spanning resource usage emulation across the entire lifetime of the simulated task. Each resource to be emulated should have an *emulator* associated with it. Each such *emulator* should implement the `org.apache.hadoop.mapred.gridmix.emulators.resourceusage .ResourceUsageEm` interface. Resource *emulators* in GridMix are *plugins* that can be configured (plugged in or out) before every run. GridMix users can configure multiple emulator *plugins* by passing a comma separated list of *emulators* as a value for the `gridmix.emulators.resource-usage.plugins` parameter.

List of *emulators* shipped with GridMix:

- Cumulative CPU usage *emulator*: GridMix uses the cumulative CPU usage value published by Rumen and makes sure that the total cumulative CPU usage of the simulated task is close to the value published by Rumen. GridMix can be configured to emulate cumulative CPU usage by adding `org.apache.hadoop.mapred.gridmix.emulators.resourceusage .CumulativeCp` to the list of emulator *plugins* configured for the `gridmix.emulators.resource-usage.plugins` parameter. CPU usage emulator is designed in such a way that it only emulates at specific progress boundaries of the task. This interval can be configured using `gridmix.emulators.resource-usage.cpu.emulation-interval`. The default value for this parameter is `0.1` i.e `10%`.

- Total heap usage *emulator*: GridMix uses the total heap usage value published by Rumen and makes sure that the total heap usage of the simulated task is close to the value published by Rumen. GridMix can be configured to emulate total heap usage by adding `org.apache.hadoop.mapred.gridmix.emulators.resourceusage .TotalHeapUsa` to the list of emulator *plugins* configured for the `gridmix.emulators.resource-usage.plugins` parameter. Heap usage emulator is designed in such a way that it only emulates at specific progress boundaries of the task. This interval can be configured using `gridmix.emulators.resource-usage.heap.emulation-interval` . The default value for this parameter is `0.1` i.e `10%` progress interval.

Note that GridMix will emulate resource usages only for jobs of type *LOADJOB*.

12 Simplifying Assumptions

GridMix will be developed in stages, incorporating feedback and patches from the community. Currently its intent is to evaluate MapReduce and HDFS performance and not the layers on top of them (i.e. the extensive lib and sub-project space). Given these two limitations, the following characteristics of job load are not currently captured in job traces and cannot be accurately reproduced in GridMix:

- *Filesystem Properties* - No attempt is made to match block sizes, namespace hierarchies, or any property of input, intermediate or output data other than the bytes/records consumed and emitted from a given task. This implies that some of the most heavily-used parts of the system - text processing, streaming, etc. - cannot be meaningfully tested with the current implementation.
- *I/O Rates* - The rate at which records are consumed/emitted is assumed to be limited only by the speed of the reader/writer and constant throughout the task.
- *Memory Profile* - No data on tasks' memory usage over time is available, though the max heap-size is retained.
- *Skew* - The records consumed and emitted to/from a given task are assumed to follow observed averages, i.e. records will be more regular than may be seen in the wild. Each map also generates a proportional percentage of data for each reduce, so a job with unbalanced input will be flattened.
- *Job Failure* - User code is assumed to be correct.
- *Job Independence* - The output or outcome of one job does not affect when or whether a subsequent job will run.

13 Appendix

Issues tracking the original implementations of GridMix1, GridMix2, and GridMix3 can be found on the Apache Hadoop MapReduce JIRA. Other issues tracking the current development of GridMix can be found by searching the Apache Hadoop MapReduce JIRA