Application-sensitive resource tuning in the SAM-Grid

The purpose of this document is to cover aspects of the SAMGrid configuration that can establish specific storage resource utilization patterns via use of pre-defined application types. Data request prioritization and data queuing along with optimal storage resource selection are tasks that help ensure consistent efficiency of the system. These tasks are particularly difficult to accomplish in the system that needs to routinely support multiple concurrently running production activities(applications). To address these tasks, SAMGrid offers means to group processing resources into independent patterns that best suit each individual production activity. These patterns are enabled by XML based language that can flexibly describe variety of storage resources, storage queues, applications and respective relationship policies in such a way resources involved in production can be used more efficiently.

SAMGrid approach is based on relating an activity/application (dzero_reconstruction, dzero_reco_merge, binary_fetch, fss_stager, etc.) to a set of configured resources (input_storage, output_storage, batch adapter handler) that such described activity/application is more efficient in using. For example, DZero's reco may be instructed to use a specific storage type for the raw data by configuring a link between the SAMGrid dzero_reconstruction application type and the input_storage element which points to the desired storage location.

As the result, we have defined several new elements in the jim_config and jim_job_managers products. In general, Jim_config is responsible for the description of site resource elements. These elements now include information extended to support variety of storage resources. The jim_job_managers configuration defines the way applications are allowed to use the resources as defined in jim_config.

The proper way of defining new elements from scratch in the JIM configuration framework is to tailor both jim_config and jim_job_managers products. In order to do so the following commands can be used: “ups configure_complex_site jim_config” and “ups tailororcomplex jim_job_managers”. If a configuration exists, it is best to revert to interactive expert mode that is available by using the jim_configure.sh tool. The tool will read the product configuration and will make the configuration text available for editing using ‘vi’. Example commands are “jim_configure.sh jim_config” and “jim_configure.sh jim_job_managers”.

**Resource configuration elements (jim_config)**

Here is an example XML coding used to describe site wide resources. That includes storage, fcp, amd fcp groups.
The "output_storage" type element defines a group of locations that may be used to place files which are generated by one of the SAMGrid applications. The "input_storage" type element defines a group of station nodes that may be used to receive incoming data (note the difference between output and input storage groups in the "location_selector_pattern" definition).

The queue group defines a set of data queues that should be used in conjunction with the selected storage.

All resource type elements share the attributes "name", "location_selector_pattern", and "location_selector_algorithm".

Resource attributes:

- "name": a string to be referenced in the jim_job_managers configuration. Must be unique.
- "location_selector_pattern", "queue_selection_pattern": Its purpose is to initialize the algorithm which will ultimately select a pair of strings for a particular application. The input to that algorithm varies by the type of
considered resource. For input_storage, fcp_queue_group or output_storage elements we have:
  o input_storage, the input text is “sam dump station -disks”.
  o output_storage, the input text is the set of all possible strings (i.e. permutations of all possible letters).
  o fcp_queue_group, the input text is the set of all fcp queue names as listed in the already-selected storage definition element.

- location_selector_algorithm. The interpretation of the <queue|location>_selector_pattern varies depending on the value of this attribute. For “random” and “local” values of the algorithm, the pattern acts as the POSIX regular expression filter. This expression filters the input as represented by plain text coming from dynamic part of the selection algorithm defined above. A similar rule applies to the “affinity” algorithm selection. The difference is in the algorithm pre-selection of the particular regular expression component based on the affinity definition (the left side of the “::” term) Among the set allowed by the pattern, single element is selected. At the moment, the selector algorithm supports only 3 modes: “random”, “local”, and “affinity”.
  o The “local” mode selects the element at the host where the application is currently running. The mode applies for fnal-farm, where all storage is local to all applications.
  o The “random” mode makes a random choice among available candidates allowed by the filter pattern.
  o The “affinity” mode enables selection based on the host name where application is running. The leftmost part of the double colon (“::”) is the affinity regexp string establishes the mapping of the matching host name to the selection pattern for the desired set of queue, input, or output storages. Thus, the final selection pattern will only apply to host names that match the leftmost part of the double colon expression. The same host name may match several affinity expressions. In such a case, the result of application of the respective rightmost parts of the affinity expressions is aggregated. In the case where the host match is not found among listed affinity expressions, the selections are made based on a random pick from all strings that match the aggregated list of alternative “default” expressions and that do not contain the double colon qualifier. All affinity expressions (whether containing the double colon or not) must be separated by a double pipe (“| |”). If that is not maintained, the parts separated by the remaining double pipe are treated as a regular storage selection expression and thus may not be classically valid.

**Application type configuration elements ( jim_job_managers).**
In addition to the applications that SAM-Grid already supports (dzero_reconstruction, dzero_reco_merge, dzero_montecarlo, etc.), two
new types have been introduced: binary_fetch and fss_stager. These new types are “sub-applications” used by dzero_reconstruction, dzero_reco_merge, etc.

The element “binary_fetch” is a placeholder to configure input storage for the DZero executables, mc_runjob, Monte Carlo card files, etc. The element “fss_stager” is a placeholder for the buffer output area used by FSS stagers.

Each application type can have input_storage and output_storage elements. Below is the XML representation for these two elements:

```xml
<input_storage name="name of the storage">
  <prot_fcp queueName="sam_fcp queue name" />
</input_storage>

<output_storage name="name of the storage">
  <prot_fcp queueName="sam_fcp queue name" />
</output_storage>
```

Both input_storage and output_storage elements may contain a prot_fcp element. This element defines the fcp queue GROUP name to throttle number of concurrent transfers to/from the respective storage. The queue group is used to resolve the actual fcp queue name on a case by case basis. Thus, all fcp queues in that group must be configured and run on all nodes that may be picked by storage selection rules defined above. Fcp queues can be configured by tailoring the sam_fcp product (see below). Note that the configuration of input_storage and output_storage alone does not enable the use of fcp. Use of the fcp queue group can only be declared in the storage reference part of application itself. If fcp_queue_group is not found in the jim_config, the name of the queue as set in the prot_fcp is used in place of the queue name itself.

These are examples of application configurations:

```xml
<dzero_reconstruction>
  <local_data_buffer>
    <input_storage name="name of the storage" />
    <output_storage name="name of the storage"/>
  </local_data_buffer>
</dzero_reconstruction>
```

The “input_storage” element defines the raw data storage location, while the “output_storage” element defines the durable location used for the DZero reconstruction application. The presence of input_storage or output_storage is optional.

```xml
<dzero_reco_merge>
  <local_data_buffer>
    <input_storage name="name of the storage" />
  </local_data_buffer>
</dzero_reco_merge>
```
The “input_storage” element defines the location for files that should be merged in the by the application. Output storage is pre-defined and is set to “enstore pnfs”, so the “output_storage” element is not allowed.

```
<binary_fetch>
  <local_data_buffer>
    <input_storage name="binary_storage" />
  </local_data_buffer>
</binary_fetch>
```

The “input_storage” element defines the storage for the DZero executable, Monte Carlo card files, etc. The application does not produce an output, so the “output_storage” element is not allowed.

```
<fss_stager>
  <local_data_buffer>
    <output_storage name="fssBuffer">
      <prot_fcp queueName="fssBuffer" />
    </output_storage>
  </local_data_buffer>
</fss_stager>
```

The “output_storage” element defines the location of the FSS stager buffer area. This area must be visible to an FSS stager. Files that are stored to durable or permanent storages are initially staged here by the job. The “input_storage” element is not needed.

The fcp configuration:
In contrast to previous releases, the new sam_fcp supports multiple fcp daemons that can run on the same host. Each daemon is named after the queue that defines the daemon port number, timeout and transport mechanism used when transferring files. In order to enable sam_fcp on the worker nodes, $SAM_CLIENT_DIR/etc/sam_cp_config.py needs to be modified to select sam_fcp as the transport protocol of choice.

This is an example of 2 fcp queues configured in Lyon, France:

```
<fcp_queue name="default">
  <fcp_port port="7788" />
  <max_xfers transfers="5" />
  <transfer_mechanism name="jim_gridftp" />
  <time_out value="3600" />
</fcp_queue>
<fcp_queue name="fssBuffer">
  <fcp_port port="7789" />
  <max_xfers transfers="3" />
  <transfer_mechanism name="jim_gridftp" />
  <time_out value="3600" />
</fcp_queue>
```
**Configuration example: the ccIN2P3 data flow**

Access to the binary input is multiplexed between HPSS and the ccsvli16 node, effectively increasing the bandwidth dedicated to binary transfers. Before the cut, this access was serialized from HPSS only. The IO load of the head node can be controlled by tuning the number of concurrent transfers for the sandbox input and recoT output.

In2p3 SAMGrid Dataflow setup (reco and merge). Before the cut.

Note: The tag next to the arrows indicates the maximum number of concurrent transfers. “Inf.” stands for unlimited.
In2p3 SAMGrid Dataflow setup (reco and merge). After the cut.

The following page shows the configuration of the site resources (jim_config), the application types (jim_job_managers), and sam_fcp. The arrows indicate the links between the site and application configurations.
<dzero_reconstruction>
  <local_data_buffer>
    <input_storage name="hpss_storage" />
  </local_data_buffer>
</dzero_reconstruction>

<dzero_reco_merge>
  <local_data_buffer>
    <input_storage name="hpss_storage" />
  </local_data_buffer>
</dzero_reco_merge>

<binary_fetch>
  <local_data_buffer>
    <input_storage name="binary_storage" />
  </local_data_buffer>
</binary_fetch>

<fss_stager>
  <local_data_buffer>
    <output_storage name="fssBuffer">
      <prot_fcp queueName="fssBuffer" />
    </output_storage>
  </local_data_buffer>
</fss_stager>

<input_storage name="hpss_storage"
  location_selector_algorithm="random"
  location_selector_pattern="rfio" />

<input_storage name="binary_storage"
  location_selector_algorithm="random"
  location_selector_pattern="ccsvli16|rfio" />

<fcp_queue name="default">
  <fcp_port port="7788" />
  <max_xfers transfers="5" />
</fcp_queue>

<fcp_queue name="fssBuffer">
  <fcp_port port="7789" />
</fcp_queue>

<output_storage name="fssBuffer"
  location_selector_algorithm="random"
  location_selector_pattern="ccd0.in2p3.fr/samgrid/jim/jim_sandbox/buffer" />

...