Siege_3.0

Siege-PWE-ELF 3.0

See also Workflow Descriptor.

What's in 3.0?

- Refactorings & additional features to accommodate:
    1. Running on IBM systems (AIX, LoadLeveler)
    2. Basic "scheduling" across systems, including:
        a. Rudimentary "match-making"
        b. Programmatic reservation requesting
            - hard: explicit start-times given to scheduler; entire workflow is scheduled "up-front"; must complete by designated time (a.k.a. "on demand")
            - soft: largely an IBM LL feature (a.k.a. "flexible"); acts like a normal job in the queue; but the reserved resources are bound to an id, not to a specific job
            - NB: in all cases other than hard reservations, workflow nodes are scheduled lazily as they become ready to run - "just-in-time" scheduling; on non-IBM platforms, we usually bypass programmatic reservation requests
- Bug fixes and improvements

What you'll need

- Pick up siege 3.0.0 for your platform from: http://otfrid.ncsa.uiuc.edu/siege
- Use the default URIs for the services and channel ("rmi://" services on forecaster, event-channel "tcp://" connection to otfrid)
- Include a property for ELF_HOME (category="environment") set to:
    - HG: /home/ncsa/arossi/elf-3.0.0
    - CO: /u/ncsa/arossi/elf-3.0.0
    - ABE: /u/ncsa/arossi/elf-3.0.0
        - (NB: when testing of this release candidate is finished, I will add an ELF_HOME setting to the Host Information Service, so you will not need to set it in your workflow unless you wish to override it.)
- Log on to Siege as usual

What has changed wrt Siege?

1. Naming and semantics of the <scheduling> properties
2. How to specify platform-specific configurations
3. <options> which you can add to the <scheduling> element
4. <global-resource> (for the <workflow>) and <resource> (for <execute> nodes) can have comma-delimited list of hosts (machine node names)
5. <resource> (for <execute> nodes) can specify resource matching via a variable name beginning with '@'
6. When you launch a workflow, you will now be asked to confirm the URI of the service to which you are launching; in Siege, each submission action auto-refreshes the client so that Siege does not hang on dead RMI connections

---

What has changed wrt the rest?

Lots of internal refactorings which you needn't worry about.

Behavior should remain largely the same. However, you may note the following:

1. The "SCHEDULED" state for the workflow has been replaced by "ACTIVE" (obviously, if we schedule lazily, as we now do in the common case, it does not make sense to say the whole workflow has been scheduled);
2. Scheduling events can be observed at the workflow level (choosing the workflow events option on the workflow summary page).

NB: if you cancel a workflow and some nodes are in a transitional state such as READY or PENDING, it may be necessary to cancel the nodes individually in order to move the workflow from "CANCEL_PENDING" (this was done to improve consistency).

SIEGE (1): <scheduling> properties

These are the properties available to be set in a profile used for <scheduling> in the workflow description. See also Job XML Schema.

There is essentially only one required property for all workflows; default is interactive ...

```
        <property name="submissionType" type="string" /><!-- batch / interactive -->
```

For batch submissions, the following are usually required, though some systems and/or queues have default settings:

```
<property name="account" type="string" />
<property name="maxWallTime" type="long" /> or
<property name="minWallTime" type="long" />
```

The following are the most common ways to request processors or cores:

```
<property name="ranksPerMember" type="int" />
<property name="threadsPerRank" type="int" />
<property name="maxCpus" type="int" />
<property name="minCpus" type="int" />
```

Explanation:

If cpus are not set, they will (both) default to the number of cpus needed to satisfy all members (=1 for non-parametric workflow nodes); ranks is the MPI parallelism, on a member-by-member basis (default = 1); threads is number of threads/cpus assignable to each rank (default = 1).

For non-mpi workflow nodes to which you wish to assign k > 1 cpu (per member), either ranks or threads can be set to k (though it perhaps makes more sense to view this as a case of rank = 1, threads = k); in any case, on non-LoadLeveler systems, only the total cpus per member has any meaning.

The following are specialized for a "master-worker" node where an (ELF) script acts as the master and launches k workers through a scheduler (as with the IBM LL version of what we call "glide-in"); these specify the submission properties (as opposed to the total request given to the scheduler for reservation purposes), as we do not want to submit the master requesting all the reserved resources (else workers will not be able to run in the reservation); note this is different from a master submitting to a "glide-in" batch system, where the main script does request the full resources which it is responsible for distributing to the workers.

```
<property name="masterRanks" type="int" />
   <property name="masterThreadsPerRank" type="int" />
```

If you wish to override the automatic computation of how many members there are in a glide-in partition (i.e., in order to grab extra work from the TupleSpace):

```
<property name="maxMembers" type="int" />
```

If you wish to override the computation of how many nodes are necessary to run the job based on the number of cores per node, use the following:

```
<property name="coreUsageRule" type="string" />
```

EXAMPLE: ABE:7,BLUEPRINT:16

where the name corresponds to HostInfo key for the machine, and the number is cores per node to be used.

The following defaults to true; it indicates that unused cores on a node allocated to this submission should be used if possible; i.e., if you are running 4 16-wide members on nodes with 32 cores, you'd ideally like to run on 2 nodes, not 4; set to false if you want to enforce only 16 per node (for LL):

```
<property name="shareNodes" type="boolean"/>
```

To indicate to the ELF container that members should not have to run as long as the container needs to:

```
<property name="maxWallTimePerMember" type="long" />
   <property name="minWallTimePerMember" type="long" />
```

These default to 'std[].log' in the initial directory:

```
<property name="stdout" type="string" />
<property name="stderr" type="string" />
```

Other optional properties:

```
        <property name="queue" type="string" />
        <property name="stdin" type="string" />
        <property name="maxTotalMemory" type="long" />
         <property name="minTotalMemory" type="long" />
        <property name="dplace-trace" type="boolean" />
        <property name="nodeAttributes" type="string" />
```

(The last one, if supported, allows you to specify a particular kind of machine node to run on.)

Optional properties that are currently unimplemented/unsupported:

```
        <property name="maxCpuTime" type="long" />
           <property name="minCpuTime" type="long" />
           <property name="maxCpuTimePerMember" type="long" />
           <property name="minCpuTimePerMember" type="long" />
        <property name="maxMemoryPerNode" type="long" />
           <property name="minMemoryPerNode" type="long" />
           <property name="maxSwap" type="long" />
           <property name="minSwap" type="long" />
           <property name="maxDisk" type="long" />
           <property name="minDisk" type="long" />
           <property name="maxBandwidth" type="long" />
           <property name="minBandwidth" type="long" />
        <property name="maxOpenFiles" type="long" />
        <property name="maxStackSize" type="long" />
        <property name="maxDataSegmentSize" type="long" />
        <property name="maxCoreFileSize" type="long" />
        <property name="checkpointable" type="boolean" />
        <property name="suspendable" type="boolean" />
        <property name="restartable" type="boolean" />
        <property name="priority" type="int" />
```

Properties defined in the Host Information Service:

If the Host Information Service defines a property for a given host (as an "environment property"), and the name of that property is also included in a scheduling profile for a workflow, then that host will only be included among the potentially matching targets if the values for that property are the same. For instance, if

`SUPPORTS_GLIDE_IN="false"`

for hostA, and a scheduling profile contains

```
               <property name="SUPPORTS_GLIDE_IN" type="boolean">
                       <value>true</value>
               </property>
```

then that property will be enforced on the match such that hostA will be excluded as a possible match for the node with that scheduling property.

SIEGE (2): platform-dependent properties

In order to allow a single workflow to run on multiple independent resources, it may be necessary to define certain properties (such as paths) according to the targeted resource. This can be achieved by doing the following:

1. Define a profile containing a platform-specific configuration as a property included within an <execution> profile (note that the category must be "platform.configuration"):

```
<execution>
    <profile name="paths">
      <property name="paths-${HOST_KEY}" category="platform.configuration"/>
    </profile>
</execution>
```

The following variables are replaced on the basis of the target resource information contained in the Host Information Service:

```
HOST_KEY:                e.g., "ABE"
NODE_NAME:                e.g., "grid-abe.ncsa.teragrid.org"
ARCHITECTURE:                 e.g., "x86_64"
OS_NAME:                e.g., "Linux"
OS_VERSION:                e.g., "2.6.9-42.0.10.EL_lustre-1.4.10.1smp"
```

2. Configurations for this profile, based on the various possible resolutions of the variable, are then written to the TupleSpace service. For example,

```
<tspace-entry-builder id="1" owner="/C=US/O=National Center for Supercomputing Applications/OU=People/CN=Albert
L. Rossi" typeName0="platform.configuration" typeValue0="paths-ABE" name="tspace-entry-paths-ABE">
   <ranOn/>
   <payload payloadType="ncsa.tools.common.types.Configuration">
      <configuration>
        <property name="ELF_HOME" category="environment">
          <value>/u/ncsa/arossi/elf-3.0.0</value>
        </property>
      </configuration>
   </payload>
</tspace-entry-builder>
```

Presumably there would be one tuple of this sort for each resource the workflow is able to run on.

Some questions ...

1. Can platform-dependent properties appear in <scheduling> profiles?
    - ANSWER: No, only in <execution> profiles. If we think of <scheduling> properties as being used to determine what target resource to use, then obviously platform-dependent properties should not be included.
2. Then why, for instance, is "account" one of the <scheduling> properties?
    - ANSWER: A contradiction. It is included simply because it is specific to running on a resource. Note that if it were placed in an <execution> profile, the workflow would still complete successfully.
3. Which of the <scheduling> properties must appear in a <scheduling> profile?
    - ANSWER: Currently, the properties needed to make a scheduling request/reservation. These include values which affect the number of cores/nodes required and the wallclock time, along with submission type, and any properties that must match Host Information environment properties. All the others could actually appear either in <execution> profiles or platform.configuration tuples if so desired.

SIEGE (3): <scheduling><options>

Along with <profile>s, an <options> element can be included in the <scheduling> section. This element has one attribute and two sub-elements, and can appear with any combination of these defined.

1. ```
   <options algorithm="random"/>
   ```

2. ```
   <options><must-terminate-by>2009/09/03 11:30:00</must-terminate-by></options>
   ```

3. ```
   <options><rules>starttime=0:00:30;cpus=0.5,walltime=2.0;cpus=0.25,walltime=4.0</rules></options>
   ```

1. The algorithm attribute defines the method used to order potential matching target resources, defining the sequence in which they will be tried. There are currently two available algorithms, one which randomly orders the target names, and the other ("static-load", the default), which contacts the machine to determine something like a "load" number on the system.
2. Including this element indicates the workflow should be treated as "on-demand" (hard, time-based reservations determined all up front for the entire graph).
3. This element establishes a set of rules to apply, in order, to the resource request issued to each potentially matching target machine when the original request fails. Currently, there are three available modifiers: `starttime`, `cpus`, `walltime`; rules are separated by a semicolon, and clauses of the rule by commas; the predicate stands for a percentage alteration of the original value or, in the case of starttime, an increment. Thus the rules tell the scheduler to try 4 times; first with the original request; then by pushing forward the start-time; then by halving the number of cpus and doubling wall time; then finally, by taking one-fourth of the cpus and increasing wall time by 4.

SIEGE (4): <global-resource>

It is now possible to provide a comma-delimited list of machine nodes on which to run the workflow.
   ```

```
<global-resource>cobalt.ncsa.uiuc.edu,tg-login.ncsa.teragrid.org,grid-abe.ncsa.teragrid.org</global-resource>
```

All such node names must be mapped in the Host Information Service. The list tells the scheduler to choose from these nodes (and only these nodes) using the ordering algorithm discussed previously. A single node name works as before (it is automatically mapped to the entire workflow). If no global resources are defined, the scheduling module chooses from all the possibly matching hosts in the Host Information's database.

NOTE the same semantics apply to individual execute nodes:

```
      <execute name="setR" type="remote">
          <resource>cobalt.ncsa.uiuc.edu,tg-login.ncsa.teragrid.org</resource>
      </execute>
```

This setting would override the <global-resource> definition for this particular node in the workflow graph.

SIEGE (5): <resource> variables

It is also possible to bind node assignments to each other using a "scheduling variable".

```
<execute name="setR" type="remote">
      <resource>@R</resource>
</execute>
...
<execute name="setS" type="remote">
      <resource>@R</resource>
</execute>
```

This variable (by convention these begin with '@' to distinguish them from actual machine node names) indicates that whatever resource has been assigned to "setR" must also be assigned to "setS".

NOTE: suppose setR gets assigned to 'cobalt.ncsa.uiuc.edu', runs, and then an attempt is made to assign setS to that node, but it fails. In this case, setS goes into the ATTEMPTED state, and is retried later.
