MCPS: Monte Carlo Processing Service

Greg Graham

Fermilab Computing Division

25-May-2005

Background

- High Energy Physics experiments often must generate huge Monte Carlo (simulated) data samples in order to be able to separate small signals from noise and detector effects.
- As collaboration size has grown (CMS ~10^3) and the expected data volume has grown (~10^{15} - 10^{16} bytes/year), so have the computing and storage needs.
  - CMS: Compact Muon Solenoid at CERN, begins ~2007
- IMHO: It has become rare for individual physicists and small groups to be able to own, organize, and manage their own computing resources in the face of this scale.
  - This motivates the approaches to Grid computing in what follows, though part of the motivation is also impetus from funding agencies to pool resources

Introduction to MCPS

- Regain individual user access to production and processing of High Energy Physics Monte Carlo data
  - When Monte Carlo data size is larger than typical desktop storage capacity or is otherwise inconvenient for the user
  - When there is complexity in generating/processing Monte Carlo data through multiple steps - generation, simulation, digitization, reconstruction, analysis
  - When efficient use of resources is a concern
- Use Cases
  - Individual user would like to generate a small amount of Monte Carlo for testing before making an “official” request.
  - Individual user would like to generate small to medium amounts of Monte Carlo (0.1 - 0.5 TB) without going through an “official” system.
  - Individual user would like to replicate an existing processing chain with or without small customizations.
- MCPS is not just for Monte Carlo!
  - To recover the small computing center look&feel with a big computing engine under the hood.

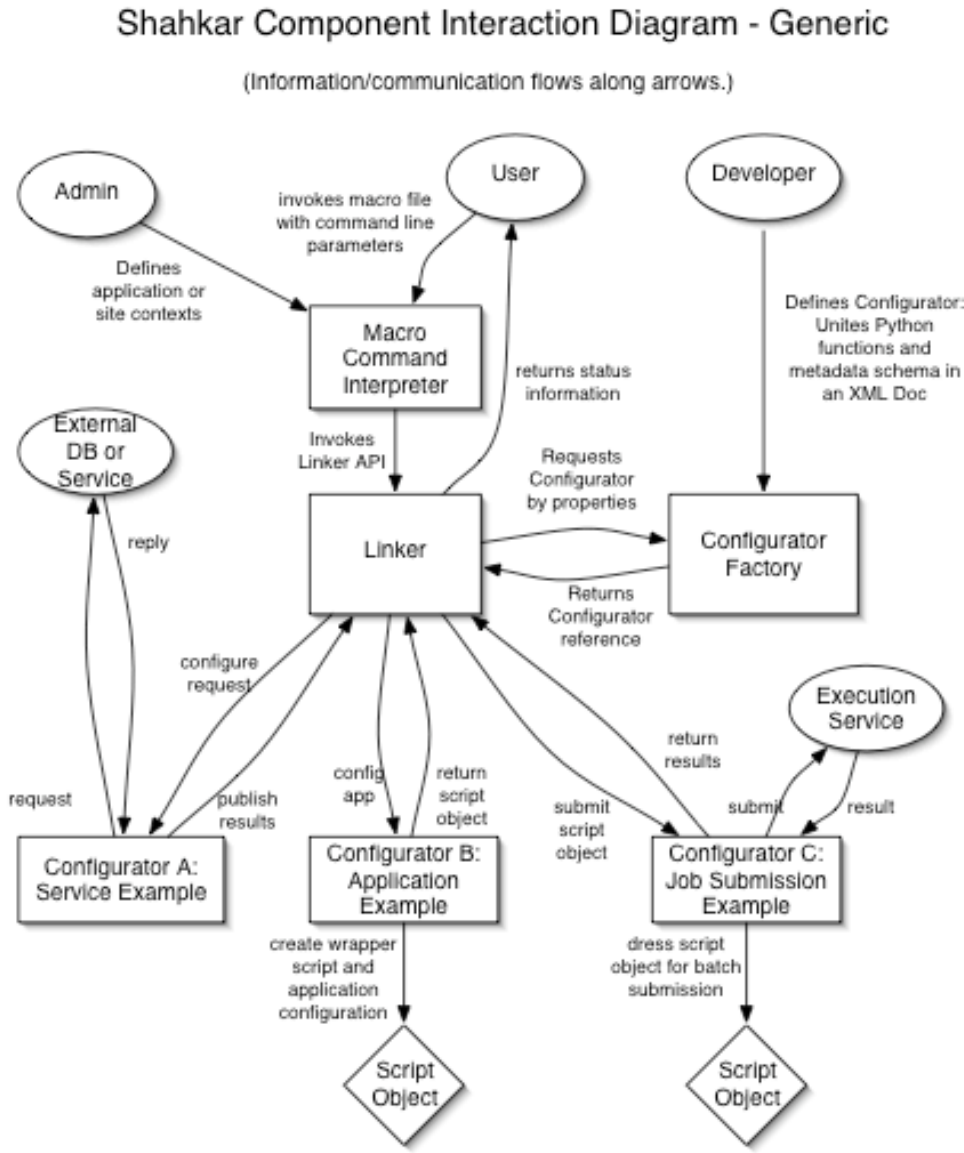
MCPS Components

- Runjob Toolkit (ShahKar)
  - Generic job creation and configuration.
  - In use by CMS underneath existing MC production tools
- ShREEK
  - ShahKar Runtime Execution Environment Kit
  - Provides instrumentation for monitoring/controlling distributed jobs
- Clarens
  - Grid-secure web services and common interfaces to data and metadata catalogs
- MCPS/Runjob
  - Job creation and configuration tailored to the CMS grid environment
- Other
  - CMS specific or Grid infrastructure components

The Runjob Toolkit (ShahKar)

- Application Configuration
  - Configures a wide variety of applications
  - Co-configures applications that may have dependencies in their parameters
- Service Configuration
  - Configures a wide variety of services
  - Interfaces to distributed computing tools and grids
- Unified Framework
  - Application and Service configurations are accessed through the same API
  - Context objects allow for configurations to be modified in a controlled way depending on logical environment

The Runjob Toolkit



Each Service or Application is represented by a distinct module called a Configurator which is solely responsible for communication with that service or configuration of executable. The Configurators exist inside of a framework called the Linker.

In response to framework calls generated by the Linker, Configurators may either do work or emit Script Objects, which are discrete tasks for later execution within jobs.

Application/Service configurations come either from user provided macros or from administrator provided context descriptions. Developers provide much of the Configurator functionality, including the schema, in XML documents.

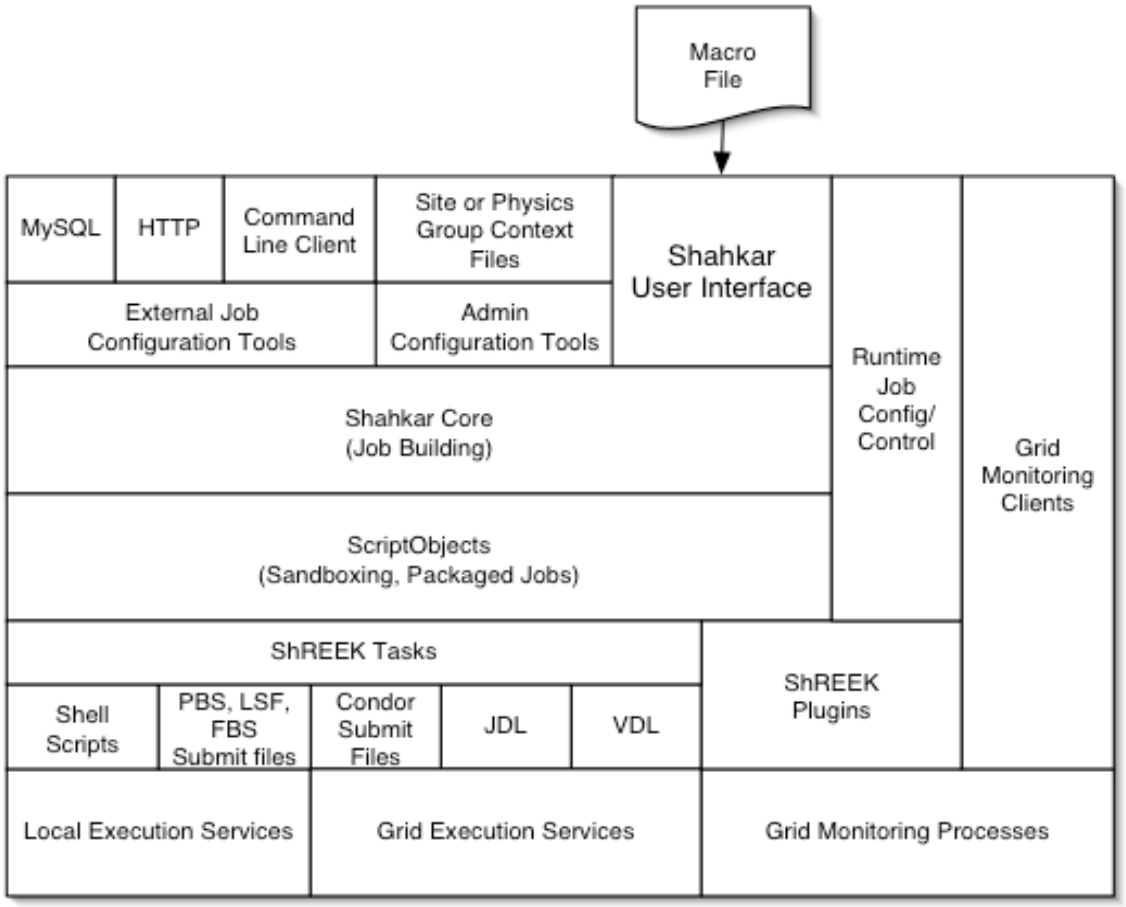
ShREEK

- Multi-threaded execution manager
  - Applications are run inside of separate processes managed by execution threads
  - A pluggable monitoring thread can run an endless variety of monitors and provide input to any monitoring system that provides an API
- ShREEK Plugins
  - Monitoring adapters: NetLogger, XML-RPC
  - Virtual shell allows users to connect to the job and do ls, cat, and simple non-destructive tasks.
- Actually ShREEK is a generic toolkit and does not in any way depend upon Runjob
  - though it works with Runjob jobs already
  - it is not yet working with Runjob “runtime” contexts, though that can be added with a plugin

ShREEK

Shahkar Architecture for Job Building and Job Submission Including ShREEK

(Information/communication flows up/down in the diagram.)



Clarens

- Web Services based client/server framework
  - Available with Java and Python bindings
- Web applications that have a Clarens wrapper benefit from:
  - Certificate based authentication and access control
  - Dynamic discovery within a cluster of Clarens servers
  - Publication of the service and method via the Monalisa framework
- Many examples and clients available
- Easy to install as root or normal user: e.g. As root do:
  - wget -q -O - http://hepgrid1.caltech.edu/clarens/setup_clump.sh
- Interoperability with other web service environments such as Globus through SOAP
- Other features
  - File system access
  - VO Management and support for Grid security model
  - Remote access to data

MCPS/Runjob

- Adds in CMS specific functionality
  - Covers CMS Monte Carlo and data processing applications
  - Provides interfaces to CMS services such as POOL
  - Provides interfaces to CMS infrastructure such as SRM
- Encapsulates complex CMS workflows
  - Invokes POOL and CMS applications transparently to build datasets
  - Handles movement of input and output data transparently
- Allows user overrides of most functionality
  - The user always has a sensible default

Other Components

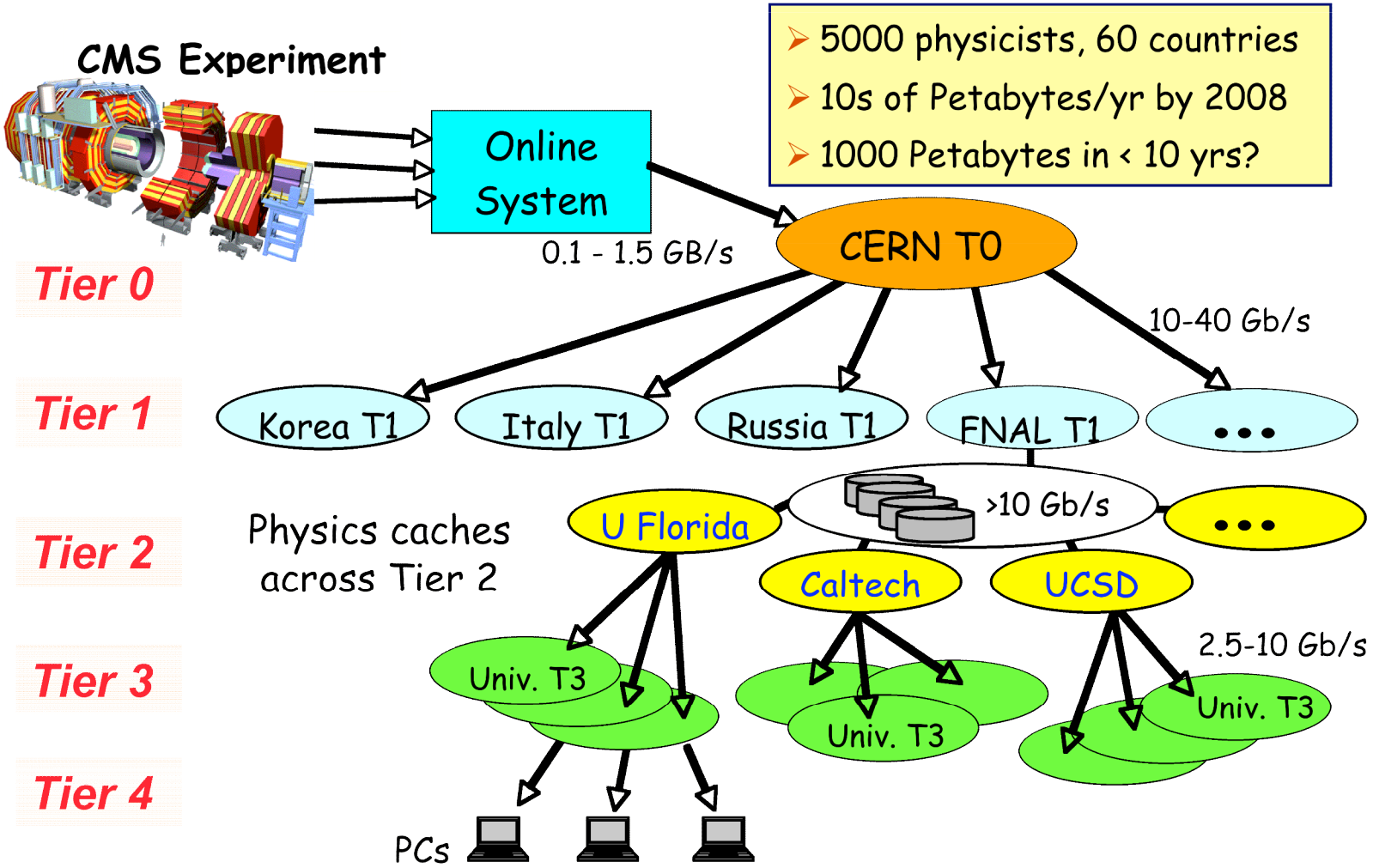
- The following components are assumed to be present as part of the infrastructure
  - MonaLisa: distributed monitoring system
    - MonaLisa is a Java based pluggable dynamic monitor
    - Works well with Clarens
  - Grid Tools: MOP, Condor-G/DAGMan (OSG flavor)
    - LCG grid tools are already working in another Runjob based Monte Carlo tool, and are easily portable to MCPS.
  - SRM/dCache/GridFTP: data storage and movement
  - BOSS: CMS application/job monitoring
    - BOSS has a wrapper executable that scans stdout for information and relays this back to a database (sometimes asynchronously)
    - It is an easily configurable application monitor (as opposed to a job monitor)

Other Components

- The following components are assumed to be present as part of the infrastructure (cont'd)
  - DAR/XCMSi/SCRAM : User application development and deployment
    - XCMSi makes RPM distributions of official CMS software releases
    - DAR can build incremental software distributions against officially released ones to capture individual software development
  - GridCat : Site local information services
  - MOP (Condor-G/DAGMan submission client)
    - (More on this later)

CMS Computing Model Overview

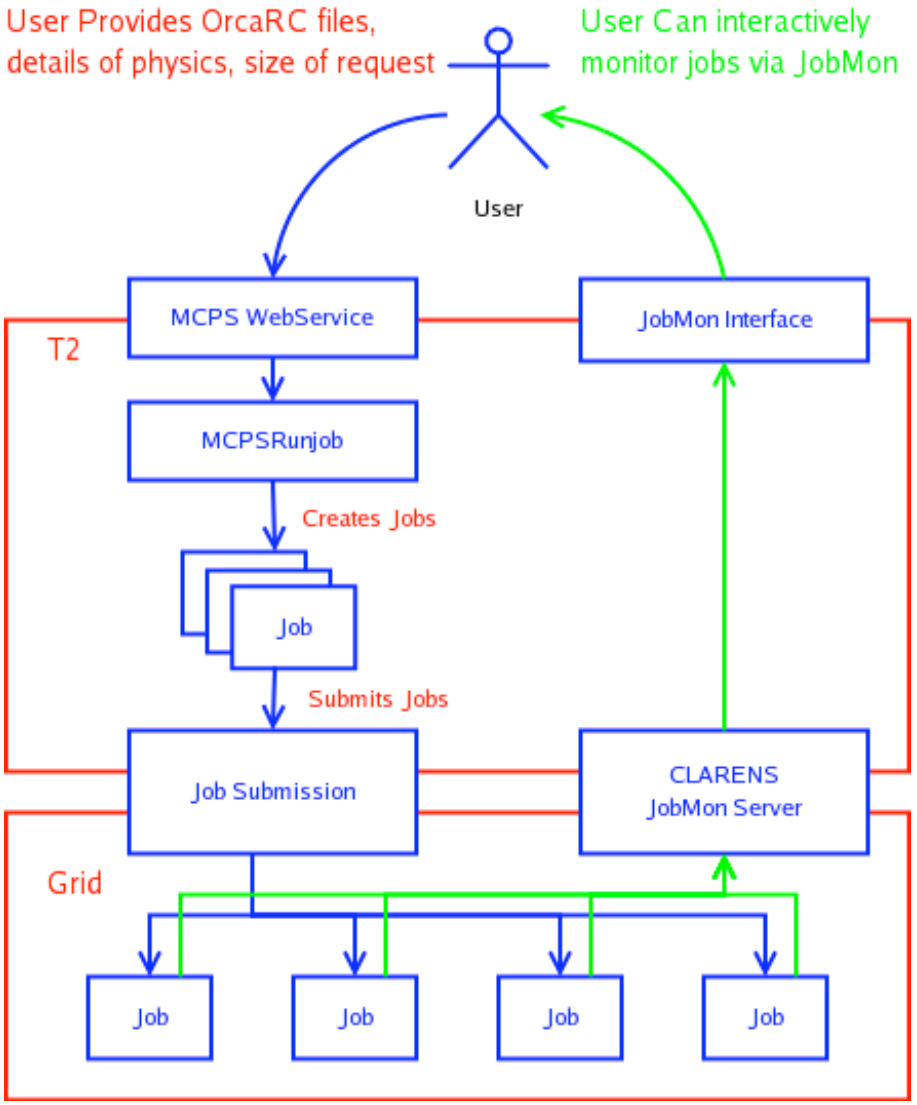
CMS Global Data Grid



MCPS Architecture Overview

- Requirements
  - Users should be able to tap into the full power of the grid as far as policy allows (ie- not technology limited)
  - Users should not be tied to a specific site
  - Users should be able to lose contact with the grid without losing their work
- Solution
  - Put up MCPS services at Tier-2 centers using Clarens
    - Use Tier-2 centers as portal to grid resources
    - Job output can be hosted by portal Tier-2 or affiliated Tier-1
    - Tier-2 hosts the job provenance
  - Some details being worked out
    - Full application monitoring system to be available only after BOSS v4 release in July.

MCPS Architecture Overview



The user specifies jobs and optionally uploads software through the MCPS web interface (powered by Clarens.) Using an invocation handle, the user monitor the job through another Clarens interface.

Not shown:
Individual jobs are wrapped by ShREEK. A ShREEK plugin will take care to feed application monitoring information to the BOSS database as it is able. If this is impossible at runtime, then the update happens asynch.

Further Architectural Details

- Job level Requirements
  - MCPS jobs should not be site-localized until they arrive at a site for execution.
    - MCPS creates site-independent jobs.
  - Sites need to advertise the information to complete the job configuration
    - Site local parameters should be administered by the respective sites
  - Job provenance must be kept
    - Job output should be “publishable” into official experiment data management
- Solutions
  - Site local parameters are pulled out of GridCat or MDS
  - Job information is kept in an XML database colocated with the job. The XML database is returned with the log information. It contains the complete provenance.

Grid Integration

- For job submissions on Open Science Grid (OSG) we plan to use MOP
  - MOP is a Condor-G/DAGMan submission client
    - It features a simple interface for choosing site to send a job or for using Condor matchmaking to choose a site.
    - MOP has standard DAG templates to support different kinds of jobs. These include references to site parameters, and these are filled in using GridCat only after the runtime site is known.
  - MOP is not integrated with BOSS
    - Integration should not be a problem if we work with the BOSS application monitoring API
  - The CMS VO is supported by Open Science Grid (OSG) services

Grid Integration - OSG

- Gatekeeper with GT2.4 GRAM compatible interface
- Authorization via Gridmapfiles or compatible GT authorization services.
  - Sites may use GUMS for dynamic mapping of user PKI certificates to local accounts, invoked from the site gatekeeper.
  - Sites may adapt SAZ to impose site authorization and access policies.
- VOs are expected to implement policies to prevent overload of the headnodes.
  - Resource allocation policy will be enforced through batch queue priorities
- SRM V1.1 Compatible interface / GridFTP
- Monitoring through the use of Ganglia, MonaLisa and MDS. The GridCat service will be used to publish the status of and information of all the sites on the OSG Grid.

Example Scenario

- User develops analysis software on laptop and packages it relative to a known standard software distribution using incremental DAR.
- User uploads job configuration information and his/her software to a favorite Tier-2 site using a Clarens web service and gets an invocation handle back.
- Clarens invokes MCPS on the Tier-2 site.
- MCPS manages job creation and submission for the user.
- Jobs are submitted through grid portals and in general do not run at the Tier-2.
  - MCPS will not do scheduling or data/job co-location. We will rely on standard mechanisms for this.
- User (possibly from a different location) can monitor job status and possibly contact the running job itself through ShREEK.
- The job collects provenance information and updates the BOSS job tracking database, possibly asynchronously.
- User (possibly from a different location) uses the invocation handle to query job status and output location.

SC2004 Prototype

- An MCPS prototype was successfully demonstrated at SC2004.
  - Using a Clarens powered web portal, participants on the showroom floor were able to submit canned MCPS jobs created at Fermilab to USCMS grid resources
  - The final output was in ROOT tuple format distributed among the participating sites. Clarens was then used to plot the results in real time.
- Proof of concept worked, but improvements need to be made
  - Used ad-hoc grid services: MCPS needs to be integrated with real infrastructure.
  - Some services, such as BOSS, were not integrated.
- This prototype was the basis for the current work.

Conclusion

- MCPS is on track to provide user-level access to grid resources in the Summer of 2005
  - The architecture combines the power of a dedicated CMS Tier-1/Tier-2 center with a simple job creation/submission/tracking interface.
  - The goal is to bring back to the user the ability to run their own Monte Carlo
    - Plus as an added bonus they get location transparency!
- MCPS is well integrated with Grid technology
  - Especially Open Science Grid and eventually LHC Computing Grid (LCG).

References

- http://http://projects.fnal.gov/runjob
- http://clarens.sourceforge.net
- http://lynx.fnal.gov/runjob/MCPS_20Wiki_20Pages
- http://www.uscms.org
- http://www.opensciencegrid.org