Kickstarting Remote Applications

Jens-S. Vöckler[1] Gaurang Mehta[1] Yong Zhao[2] Ewa Deelman[1] Mike Wilde[3]

[1] *University of Southern California*
Information Sciences Institute
4676 Admiralty Way Ste 1001
Marina Del Rey, CA 90292
`{jens,gmehta,deelman} at isi dot edu`

[2] *University of Chicago*
Dept. of Comp. Science
1100 East 58th Street
Chicago, IL 60637 `yongzh at cs dot uchicago dot edu`

[3] *Argonne National Laboratories*
Math. and Comp. Sci. Division
9700 South Cass Avenue
Argonne, IL 60439
`wilde at mcs dot anl dot gov`

The *kickstart* executable is a light-weight program that is distributed as part of the GriPhyN Virtual Data System. It sits in between the remote scheduler and the executable, gathering additional information about the executable run-time behavior, including, but not limited to, the exit status of jobs and information to ease grid debugging, in a uniform fashion without assistance from the scheduling system. *Kickstart* provides the necessary information required by provenance tracking systems to accurately annotate data products. The authors believe *kickstart* is of value for anyone running applications on the Grid. We would like to make the community aware of its existence and benefits.

1 Motivation

The GriPhyN [1] Virtual Data System [2] (VDS) provides a set of tools for expressing, executing, and tracking the results of scientific workflows. It ships with a set of light-weight compiled C tools and scripts to be used at the remote end of workflows. Each tool enhances Globus's [4] capabilities, with *kickstart* offering the most potent extensions.

When multiple jobs and dependecies among them are tied into a workflow, knowledge about the success or failure of finishing jobs is required before dependent jobs can be started. Besides the exit code, signals are important part of determining success or failures. For instance, some applications use assertions, causing an abnormal termination with SIGABRT when these are violated.

In an environment of pre-WS GRAM services, available in all releases of the Globus® Toolkit, information about the success or failure of remote application executions is not returned to the submitting entity. On the other hand, WS-GRAM [5] returns the remote exit code of the application or which signal an application may have died upon, if the underlying scheduler supports passing it.

The original objective for *kickstart* was to reliably provide the remote application's complete exit status, the exit code and any signals, in order to drive a workflow. It has since evolved into an extensible tool for job monitoring, Grid execution debugging and provenance tracking. *Kickstart* is able to gather additional information such as the resource usage of the application that just finished. Further features were added over time, collecting information about job status, file stat records, details about the compute environment etc, which are required by a provenance tracking systems. Features are discussed in section 3.

2 Architecture

Kickstart was designed to mesh well with the Globus environment, both pre-WS and WS-GRAM. *Kickstart* starts applications, not web services. The design, originally focussed on obtaining the remote exit status, is not intended to present additional challenges. To overcome firewall issues, *kickstart* takes over the *stdio* of the grid job, and lets Globus deal with the firewall setup to tunnel *stdio* back to the submit host.

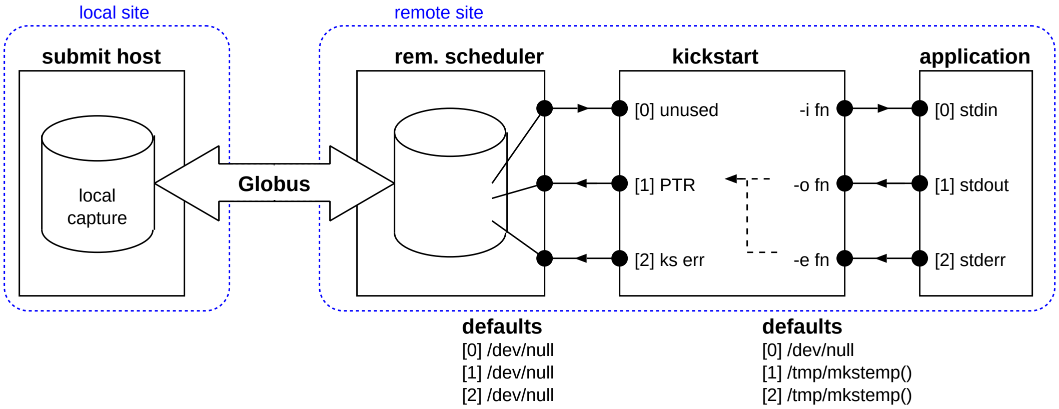


Figure 1: *Kickstart*'s position in the grid execution pipeline.

Figure 1 shows the architecture where *kickstart* sits between the scheduler and the application. The application's *stdio* are controlled by command-line flags, and attached accordingly to files or devices. However, if a user does not indicate a file, the application's *stdout* and *stderr* are still captured in temporary local files to aide grid debugging. A configurable amount of information from these files becomes part of the result record *kickstart* produces.

Kickstart currently passes the grid job's *stdin* transparently to the application. The Pegasus [3] planner exploits this feature to pass configuration files to remote gridftp helpers. However, in the future of *kickstart* a job description will be carried on the grid job's *stdin*, to ease the proliferation of command-line options and prevent exceeding resources like the arguments length limit.

The *stdout* of the grid job provides the provenance tracking record (PTR) of the application's execution and current execution environment. The PTR is an XML instance document[1] which provides all data gathered, written as *kickstart* finishes.

Kickstart uses the grid job's *stderr* to report abnormal start-up conditions, and once it is running, to provide a best-effort application feedback channel. A *kickstart*-aware application will find in its environment a reference to the feedback channel, upon which the application can send small data records to the submit host.

While *kickstart* employs GRAM to transport data, it is independent of Globus. *Kickstart* can be utilized locally, in shell scripts and directly on the command-line.

3 Features

Kickstart primarily captures the remote application's full exit status, because it reaps the processes it started. The full exit status is a 16 bit integer on many systems. The most significant byte is the exit code that an application returns by means of the `exit` system call. The least significant byte contains the signal an application may have died upon, and if a core file was generated.

```
<status raw="0">
 <regular exitcode="0"/>
</status>
```

Figure 2: XML fragment for exit status.

Kickstart also captures the resource usage of an application in terms of the system time and user time, minor and major page faults, swap operations and received signals (not supported by all

[1] The corresponding XML schema is documented at `http://vds.isi.edu/docs/`, see latest "IV" schema.

systems), as well as OS context switches. Other resource information like resident set sizes and IO operations are not supported by all operating systems, and are thus excluded for now.

```
<usage utime="0.000" stime="0.004"
 minflt="223" majflt="0" nswap="0"
 nsignals="0" nvcsw="21" nivcsw="1"/>
```

Figure 3: XML fragment for r-usage record.

Along with the resource usage for each application, *kickstart* captures information about the start wall-time according to the host's clock, the job duration wall-time, the host's primary interface's IP address, the user-id, group-id, the current working directory, the environment variables a process sees, and uname-style information about the platform itself.

Kickstart will connect the application's *stdio* file descriptors to user provided files. If a user does not care about the application's *stdout* or *stderr*, *kickstart* will still capture these into temporary files. A configurable chunk of both files will be placed in the PTR, permitting grid debugging. For instance, if a remote application fails to run properly, it often sends warnings to *stderr*. By always capturing and returning both, *stdout* and *stderr*, *kickstart* eases grid debugging, enabling a user at the submit host to see what is going wrong. This feature is meant for debugging only. If any significant data is produced on *stdout* or *stderr*, these should be tracked data product files.

The current version of *kickstart* is set up to reflect some data records from the submit side into the PTR, notably the logical name of the application and the site handle, an unique identifier for the site.

When *kickstart* starts, it creates a named pipe (FIFO) in the local filesystem, and places the name of the FIFO in the environment of any job that it starts. Thus, by dereferencing an environment variable, a *kickstart*-aware application can send arbitrary textual data to the submit side by writing to the FIFO. It even works in shell script jobs.

Many workflows use Condor-G [9] to add a layer of reliability and restartability over the Grid. However, Condor places a stringent limit of 4000 characters on the length of the arguments, much lower than the customary 64k and 128k offered by most systems. To overcome this limit, *kickstart* can read its arguments from a file. This file can be easily shipped from the submit host using appropriate Condor directives.

Kickstart obtains a `stat` record for files such as the application executable that is to be started, *kickstart* itself, all *stdio* files and the FIFO. With help of the stat information, application failures due to insufficient rights or staging failures, resulting in zero-sized files, can be detected postmortem, even if the remote job directory no longer exists.

```
<statcall error="0">
 <file name="/bin/date">...</file>
 <statinfo mode="0100755" size="47684"
  inode="3744796" nlink="1" blksize="4096"
  mtime="2005-07-25T08:39:46-07:00"
  atime="2006-09-11T15:20:01-07:00"
  ctime="2006-03-30T04:27:51-08:00"
  uid="0" user="root" gid="0" group="root"/>
</statcall>
```

Figure 4: XML fragment for stat call.

The information record above shows that the stat call worked fine (The "error" attribute is 0). The operand was the executable `/bin/date`, shown abbreviatedly. The result record of the `stat` call includes the file access mode in octal representation, the file's size, its inode number on the filesystem, the link count for the file, the number of disk blocks it occupies, and the file's owner and group, both numerically and symbolically.

Furthermore, *kickstart* permits one to obtain `stat` records of arbitrary files, either before the application is started, or after the application finished. Thus, it can be determined, if an input file failed to stage, has zero length, or if output was improperly produced.

Kickstart is capable of running more than just a single application. It has a notion of a setup-, pre-, main-, post- and cleanup job. If a setup job exists, it is always run before anything else. Its exit status does not matter. The pre-, main- and postjob are logically chained. The chain is broken, if any of the jobs in the chain fails, as expressed by the job's full exit status. The cleanup job is always run after all other jobs, and independent of any previous exit status.

To simplify job specifications, *kickstart* supports variable rewriting using environment variables. `${appdir}/bin/myapp` can be used as a valid notation for an application – e.g. in a transformation catalog entry – provided the environment supplies the proper `appdir` setting, which could be adjusted from a site catalog or by the site itself.

While it is considered safer to always use a full absolute path to start an application, for the shell users among us, *kickstart* will search the PATH when it starts an application with a relative path name to the executable.

Kickstart has been used for the last four years, having reached good maturity and stability in millions of executions in diverse execution environments.

4 Provenance

Provenance refers to the derivation history of a data product, from its origination, up to its current state. *Kickstart* captures the information about how a data product was created, and the resources and the execution environment used to produce it.

Some applications access files which are mentioned neither in configuration nor invocation of the application. For this reason, *kickstart* makes no assumptions about data products. By design, it has no knowledge about them. The assocation between data products and *kickstart*-provided metadata is up to the provenance system. *Kickstart* provides necessary, accurate and essential, albeit not sufficient, information that is generally associated with data product provenance.

Kickstart's information is obtained in very close temporal and spatial proximity to the application being executed. Thus, its approach is superior to gathering the same necessary provenance information from other sources of less proximity. External services and even on-site catalogs always have a chance of being out of date, misconfigured, unavailable, or simply wrong. Not all services have the same access to all the information that *kickstart* is able to gather as concurrent parent of processes it starts.

Most of the data provided by *kickstart* can be used to annotate data products, or associated with a workflow and stored in the provenance tracking catalog (PTC) of the VDS. The history data in PTC, combined with recipes (workflow and application definitions) and metadata annotations, enable powerful query capabilities that help discover, understand, validate and reuse data products and their associating applications and workflows [6].

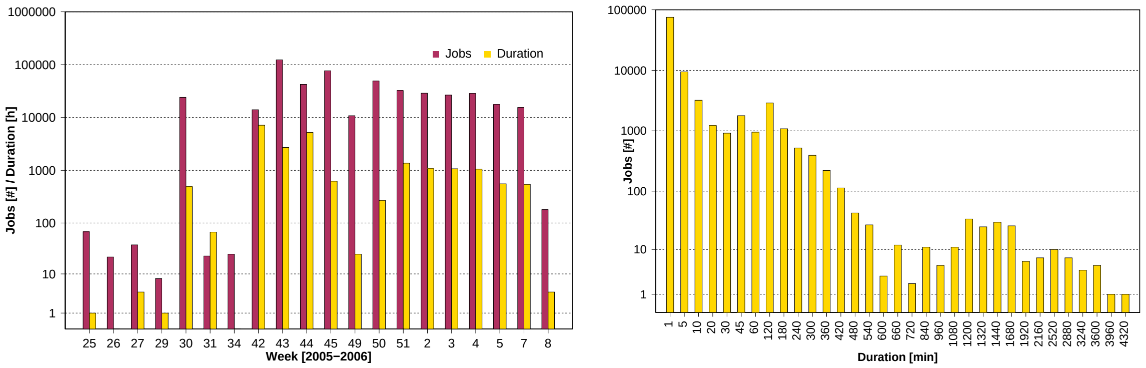
Querying the PTC itself also reveals intriguing facts about ongoing research. We present some examples in the next section.

5 Applications

CPU years	Jobs	Month
0.45	1402	2004-06
19.88	13267	2004-07
33.88	20678	2004-08
40.06	20229	2004-09
15.21	30833	2004-10
14.95	34591	2004-11

Table 1: U.S. ATLAS late 2004 jobs.

During the second half of 2004, US ATLAS [8] production was captured in a PTC kept at BNL. Grid submission points at various locations across the country contributed provenance tracking records (PTRs) in a distributed manner. Ta-



(a) Jobs and Durations over Time.

(b) Histogram of job durations.

Figure 5: SCEC Provenance Traces.

ble 1 shows the CPU-years[2], successful research domain jobs and month. Jobs were run on over 1376 distinct machines in the Grid, with the true number higher due to the fact that many sites use virtual private network addressesfor the primary interface of a worker node. While all architectures were i686 Linux systems, jobs were run on 37 different flavors thereof. In the time frame shown, over 300,000 *kickstart* records were captured.

week	start of week	CPU days
22	2006-06-03 11:56	7.69
23	2006-06-05 00:02	70.10
24	2006-06-12 09:37	234.38
25	2006-06-19 00:00	374.24
26	2006-06-26 00:44	310.23
27	2006-07-04 00:16	275.77
28	2006-07-10 00:00	163.70
29	2006-07-17 00:05	206.89
30	2006-07-24 11:26	241.35

Table 2: CPU consumption during GALE and NIST evaluations.

ISI participated in the GALE and NIST [7] evaluations of its natural language processing (NLP) system in 2006. Applications of the ISI NLP decoding workflows were run on the Grid and *kickstart*ed. Table 2 shows the data extracted from the PTC in preparation for, and during the evaluations. The data indicates that close to one CPU year was used every week. Similarly, in June and July, about 2.5 CPU-years were used respectively.

The Southern California Earthquake Center (SCEC) ran workflows as part of the CyberShake Project [10] from Sept 2005 until Aug 2006. The CyberShake project is an analysis designed to compute probabilistic seismic hazard curves for regions in the Southern California area. Hundreds of hazard curves are required to compute the final hazard map for the Los Angeles region.

Workflows using the Pegasus planner ran for eleven regions – with two regions repeated – totalling over 500,000 jobs. Of these, approximately 250,000 jobs were application domain jobs. The remaining 250,000 jobs were in the workflow management domain, including data transfer and replica registration jobs. CyberShake used about 2.5 CPU-years on two TeraGrid clusters at NCSA and Sandiego, as well as the HPC Cluster at USC. Figure 5(a) shows the number of jobs and their duration over the weeks between 2005 and 2006. Figure 5(b) shows the histogram

[2] Using 31,557,600 s = 365.25 days – the average length of a year in the Julian calendar.

of the application domain's main job's duration. Due to variations in the amount of data, a job's duration ranges between one minute and three days.

The provenance record provided by *kickstart*-enabled SCEC to estimate future computational requirements. The records also revealed how much time the code spent on different machines, the number of page faults and other useful instrumentation. Equipped with this knowledge, the SCEC scientists were able to reduce the computational requirements three to four times. *Kickstart* helped provide information about number of job failures (38,235 jobs) and their duration (2.5 hours total) due to various data transfer issues. It also helped debugging these transfer issues by capturing remote error information and returning it to the submit host for post-mortem analysis.

The examples above focus on the job durations, though the team of *kickstart* and PTC provide more than that. For instance, they can well support queries and statistics such as jobs that ran on specific sites, specific hosts, with specific OS types, the effect of different parameter values to the run times, abnormal jobs that ran a lot longer than the average run time, and so on. Any general provenance tracking scheme requires knowledge about the what, when, where, and how data products were derived. The information provided by *kickstart*, and stored in the PTC, enable provenance systems to gather these essential attributes about data products.

6 Short-comings

Kickstart's development is requirement driven. It is designed for applications, not web services. While many may consider this a short-coming, the majority of research algorithms are found in applications.

If applications want to use check-pointing, and are compiled for the Condor *standard* universe, Condor prohibits certain system calls that are required for proper functioning of *kickstart*. To solve this problem, *kickstart* needs to be integrated into a variant of the Condor `condor_startd`.

When gathering the resource usage of terminated applications, Linux and Mac OS do not provide values for all data fields. Many fields of interest are always zero. For this reason, the resource usage record in the PTR is more limited than desirable. A solution for Linux requires changes at the kernel level. It is conceivable to obtain a subset of the missing data from monitoring of the underlying system.

Many remote cluster sites are optimized for MPI support. Some experiments have shown that *kickstart* can be used in MPI environment to obtain application status without special compilation. However, it requires minor fixes to alleviate write-after-write conflicts when multiple parallel instances write their PTR into the same file.

Kickstart streams data from the application feedback channel. These information chunks can be returned during the lifetime of a job. While Globus pre-WS GRAM, by default, streams the application feedback channel data on a best-effort basis, it can only stream what the scheduler and file system make available. Due to performance and resource issues, WS-GRAM prefers to stage *stderr*. Thus, the application feedback channel feature as provided by *kickstart* is not very useful for immediate data like heartbeats, because data is returned to the submit host only after the job finished.

While it is useful to obtain and return core dumps, doing so presents some challenges. First, systems generate core dumps with different file naming strategies. Similarly, differences between the submit host and the remote execution environment, e.g. version of libc and other libraries, further limit the usefulness of inspecting a core file on the submit host. Remote resource limits may truncate the core file. Finally, files of the potential size of core dumps should not be streamed via the GASS transport which kickstart employs for its own comparatively short messages.

7 Future

Kickstart in the current version is a well-matured and stable application that has proven its merit in millions of invocations.

At some point, it is conceivable that *kickstart* will start applications in an instrumented fashion, accumulating knowledge about any `open` system call an application issues. Thus, *kickstart* will be able to automatically provide information about any file utilized by an application.

Users have frequently requested to augment the `stat` information record in the PTR with an optional MD5 sum of the file's content. A prototype exists in v2.

For some years now, a prototype of a version 2 has been used to experiment with a configuration driven instead of commandline driven application start. The configuration driven approach will alleviate a lot of the messiness concerning quoting and shell variable expansion.

Since *kickstart* is active concurrently with the job it started, it is possible to extend it further to include system resource monitoring, heartbeats, and even an API to remotely check, suspend and resume applications. A prototypical heartbeat is part of v2.

To implement the job control API above, *kickstart* needs to "phone home" by contacting a service, providing a bi-directional communication link. The service can be hosted on the submit host or on a dedicated machine. Over this link, data chunks from *kickstart*-aware applications can be multiplexed with other data, including multiple channels per application for different purposes. To overcome firewall issues, it will be necessary to run a proxy application for the phone-home protocol on the gatekeeper host.

8 Thanks

This work is supported in part by the National Science Foundation GriPhyN project under contract ITR-086044.

Our thanks go to: Ian Foster, Carl Kesselman, Mei-Hui Su, Karan Vahi, Valerie Taylor, Seung-Hye Jang, Rob Gardner, Marco Mambelli, Jerry Gieraltowski, Xin Zhao, Luc Moreau, Daniel Marcu and Steve DeNeefe.

References

[1] "The GriPhyN Project: Towards Petascale Virtual Data Grids"; Paul Avery, Ian Foster; Technical Report GriPhyN-2001-15, `http://www.griphyn.org`, 2001.

[2] "The Virtual Data Grid: A New Model and Architecture For Data-Intensive Collaboration"; Ian Foster, Jens Vöckler, Mike Wilde, Yong Zhao; in *First Biennial Conference on Innovative Data Systems Research*, Jan 2003.

[3] "Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems"; Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, Daniel S. Katz; *Scientific Programming Journal*, 13(3):219-237, 2005.

[4] `http://www.globus.org/` "The Grid: Blueprint For A New Computing Infrastructure"; Carl Kesselman, Ian Foster, editors; Morgan Kaufmann, 2004.

[5] "Globus Toolkit Version 4: Software for Service-Oriented Systems."; Ian Foster; *IFIP International Conference on Network and Parallel Computing*; Springer-Verlag LNCS 3779, 2-13, 2005.

[6] "Applying the Virtual Data Provenance Model"; Yong Zhao, Mike Wilde, Ian Foster; *International Provenance and Annotation Workshop (IPAW '06)* 2006.

[7] Working Notes of the NIST MT Evaluation Workshop; Washington D.C., September 2006.

[8] "The Capone Workflow Manager"; M. Mambelli et al.; *Proc. Computing in High Energy and Nuclear Physics (CHEP '06)*, 2006.

[9] "Condor-G: A Computation Management Agent for Multi-Institutional Grids"; J. Frey, T. Tannenbaum, I. Foster, M. Livny, S. Tuecke; *Cluster Computing*, 5(3):237-247, 2002.

[10] "Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance Tracking: The CyberShake Experiment"; Ewa Deelman et al.; 2nd *IEEE International Conference on e-Science and Grid Computing*, 2006.