the globus project™
www.globus.org

Data Management

Globus Toolkit™ Developer Tutorial

The Globus Project™

Argonne National Laboratory
USC Information Sciences Institute

http://www.globus.org/

Data Management Services

- Data transfer and access
  - **GASS:** Simple, multi-protocol file transfer tools; integrated with GRAM
  - **GridFTP:** Provides high-performance, reliable data transfer for modern WANs
- Data replication and management
  - **Replica Catalog:** Provides a catalog service for keeping track of replicated datasets
  - **Replica Management:** Provides services for creating and managing replicated datasets

GASS
Remote I/O and Staging

- Used by GRAM to:
  - Pull executable from remote location
  - Move stdin/stdout/stderr from/to a remote location
- Access files from a remote location

What is GASS?
Global Access to Secondary Storage

(a) GASS file access API

- Replace open/close with globus_gass_open/close; read/write calls can then proceed **locally**

(b) RSL extensions

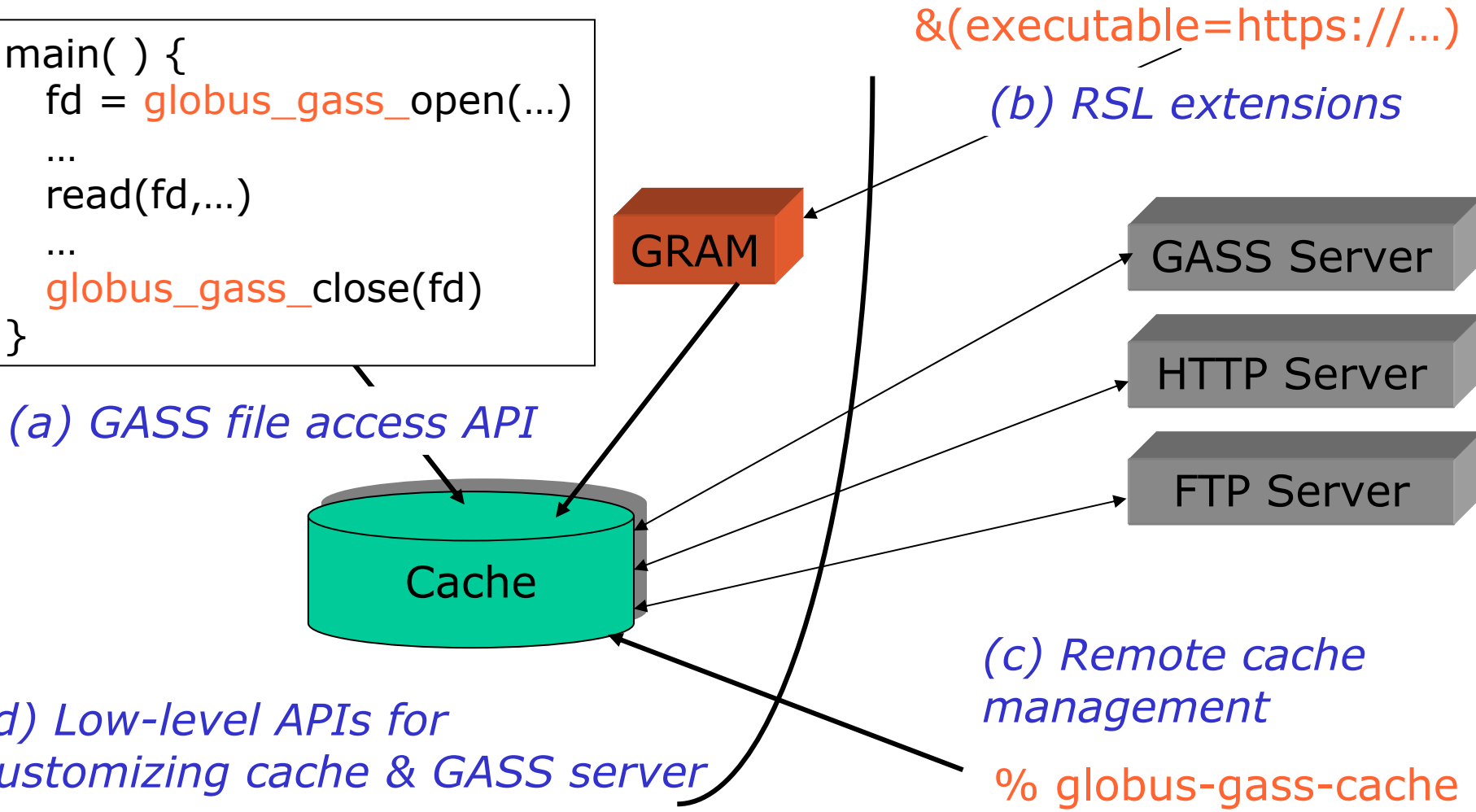
- **URLs** used to name executables, stdout, stderr

(c) Remote cache management utility

- **File cache:** a local secondary storage area in the execution machine where copies of remote files are stored

(d) Low-level APIs for specialized behaviors

GASS Architecture



GASS File Naming

- URL encoding of resource names

  https://quad.mcs.anl.gov:9991/~bester/myjob

  protocol server address file name

- Other examples

  https://pitcairn.mcs.anl.gov/tmp/input_dataset.1

  https://pitcairn.mcs.anl.gov:2222/./output_data

  http://www.globus.org/~bester/input_dataset.2

- Currently supports http & https
- Future release will also support ftp & gridftp.

GASS RSL Extensions

- executable, stdin, stdout, stderr can be local files or URLs
- executable and stdin loaded into local cache before job begins (on front-end node)
- stdout, stderr handled via GASS append mode
- Cache cleaned after job completes

GASS/RSL Example

```
&(executable=https://quad:1234/~/myexe)
  (stdin=https://quad:1234/~/myin)
  (stdout=/home/bester/output)
  (stderr=https://quad:1234/dev/stdout)
```

Example GASS Applications

- On-demand, transparent loading of data sets
- Caching of (small) data sets
- Automatic staging of code and data to remote supercomputers
  - GridFTP better suited to staging of large data sets
- (Near) real-time logging of application output to remote server (GRM-PROVE can use it)

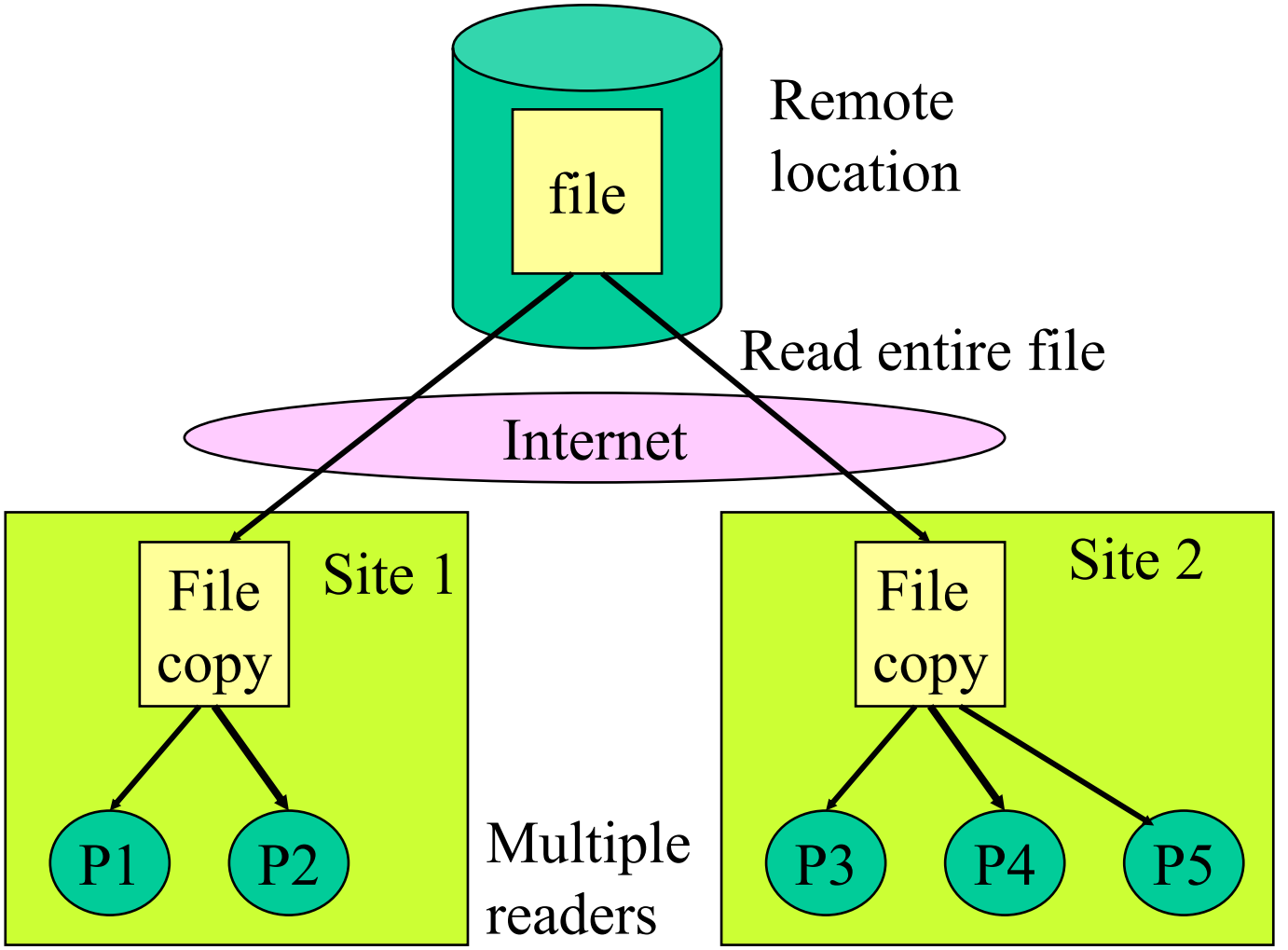
GASS File Access API

- Minimum changes to application
- globus_gass_open(), globus_gass_close()
  - Same as open(), close() but use URLs instead of filenames
  - Caches URL in case of multiple opens
  - Return descriptors to files in local cache or sockets to remote server

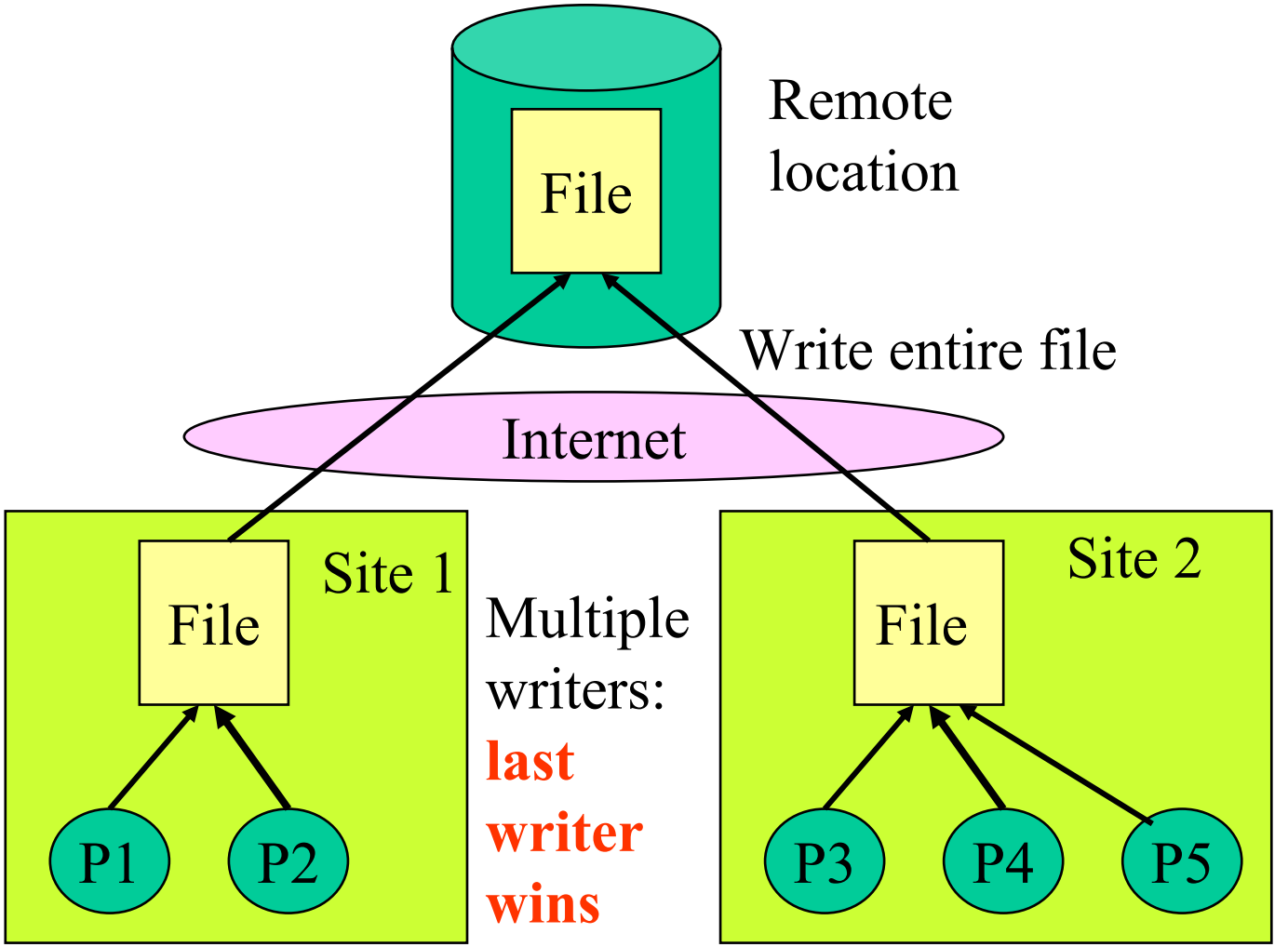
GASS File Access API (cont)

- Support for different access patterns
  - Read-only (from local cache)
  - Write-only (to local cache)
  - Write-only, append (to remote server)
- In all cases the general assumption: there is no concurrent file access among several application programs

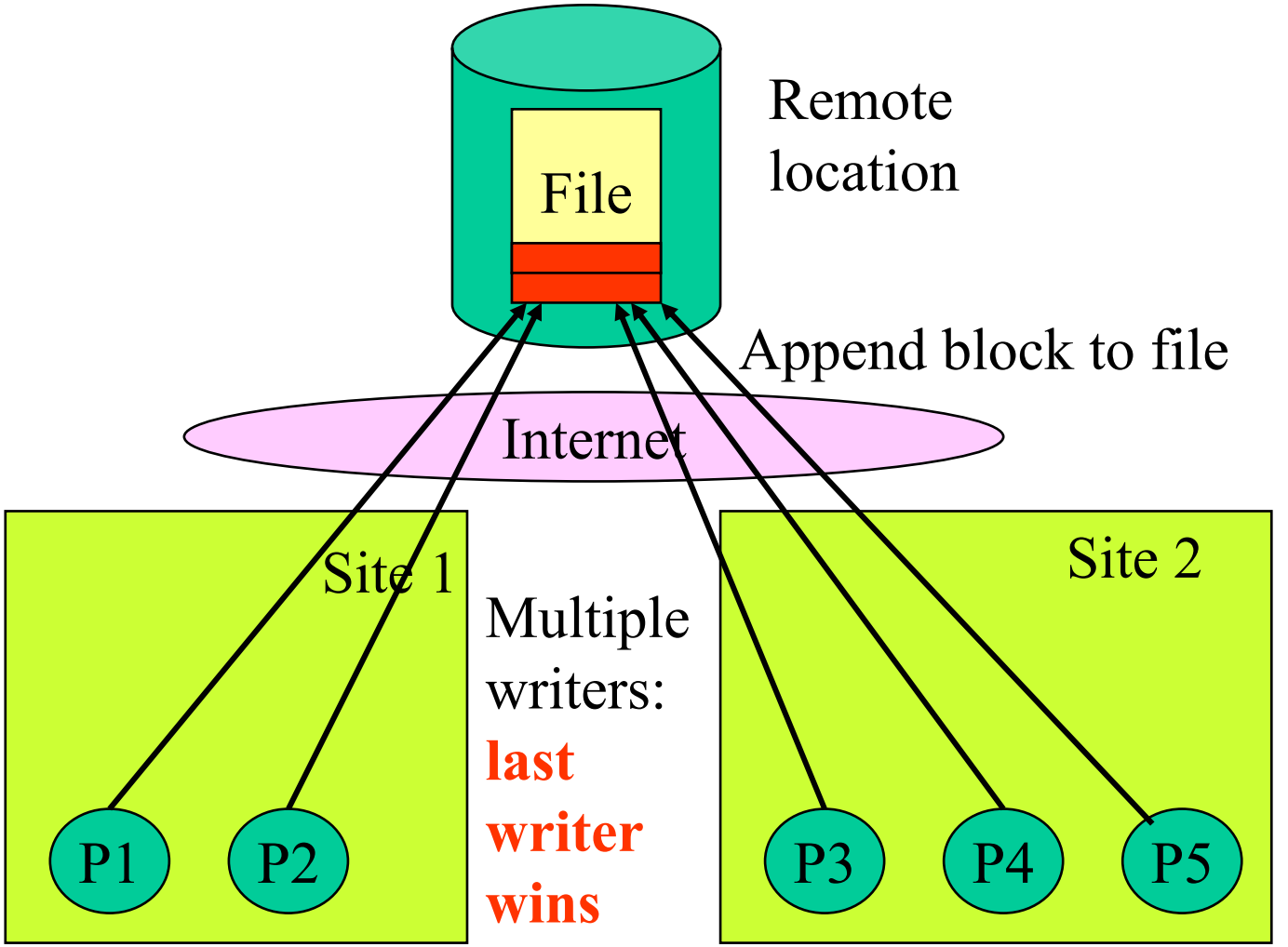
Read-only access



Write-only access



Append-only access

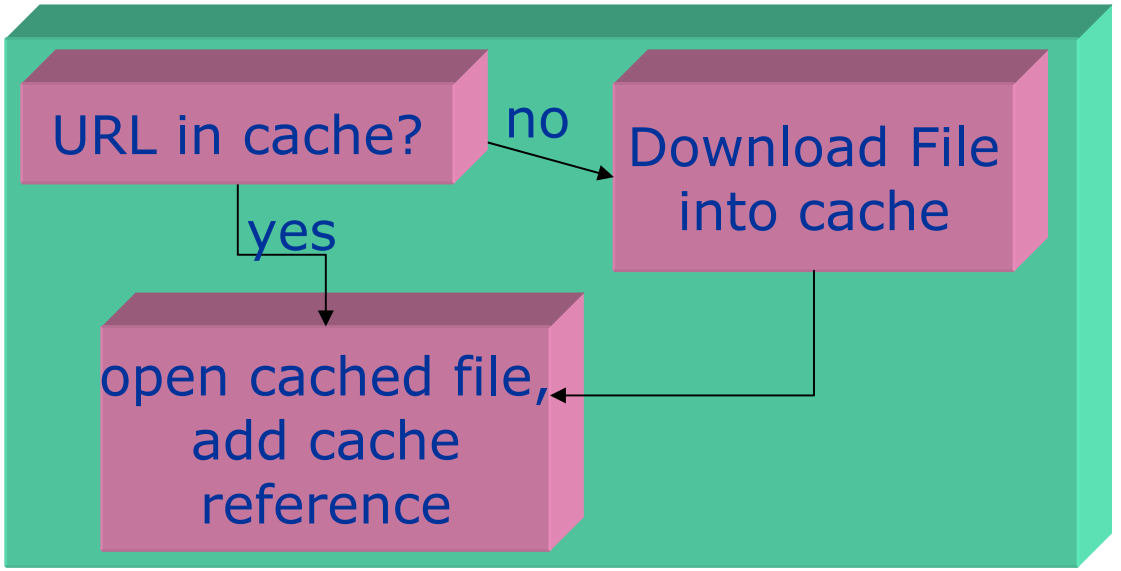


GASS File API Semantics

- Copy-on-open to cache if
  - not truncate or write-only append *and*
  - not already in cache
- Copy on close from cache if
  - not read-only *and*
  - no other copies open
- Multiple globus_gass_open() calls share local copy of file
- Reference counting keeps track of open files
- Append to remote file if write-only append: e.g., for stdout and stderr (GRM-PROVE can use it)

globus_gass_open()

- Strategy: Fetch and cache on first read open



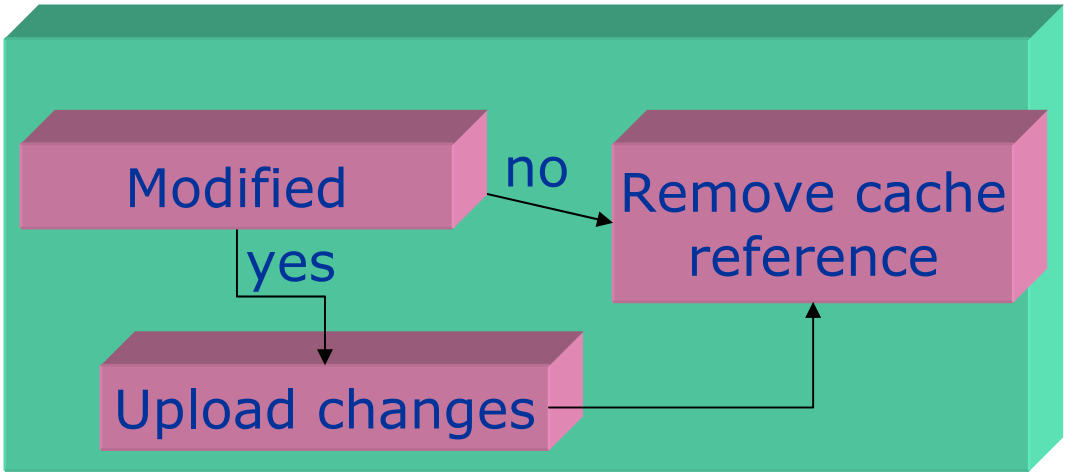
Optimized for parallel processing where several processes access the same file

GASS File Open

- Advantage:
  - The file is transferred only once even if it is used by several processes
  - The file in cache can be accessed locally by conventional I/O calls
- Disadvantage if the file is too large:
  - Computation maybe delayed too long
  - Local cache maybe too small to store the entire file
- Solutions:
  - Prestaging
  - Specialized GASS servers

globus_gass_close()

- Strategy: Flush cache and transfer on last close (for a write file)



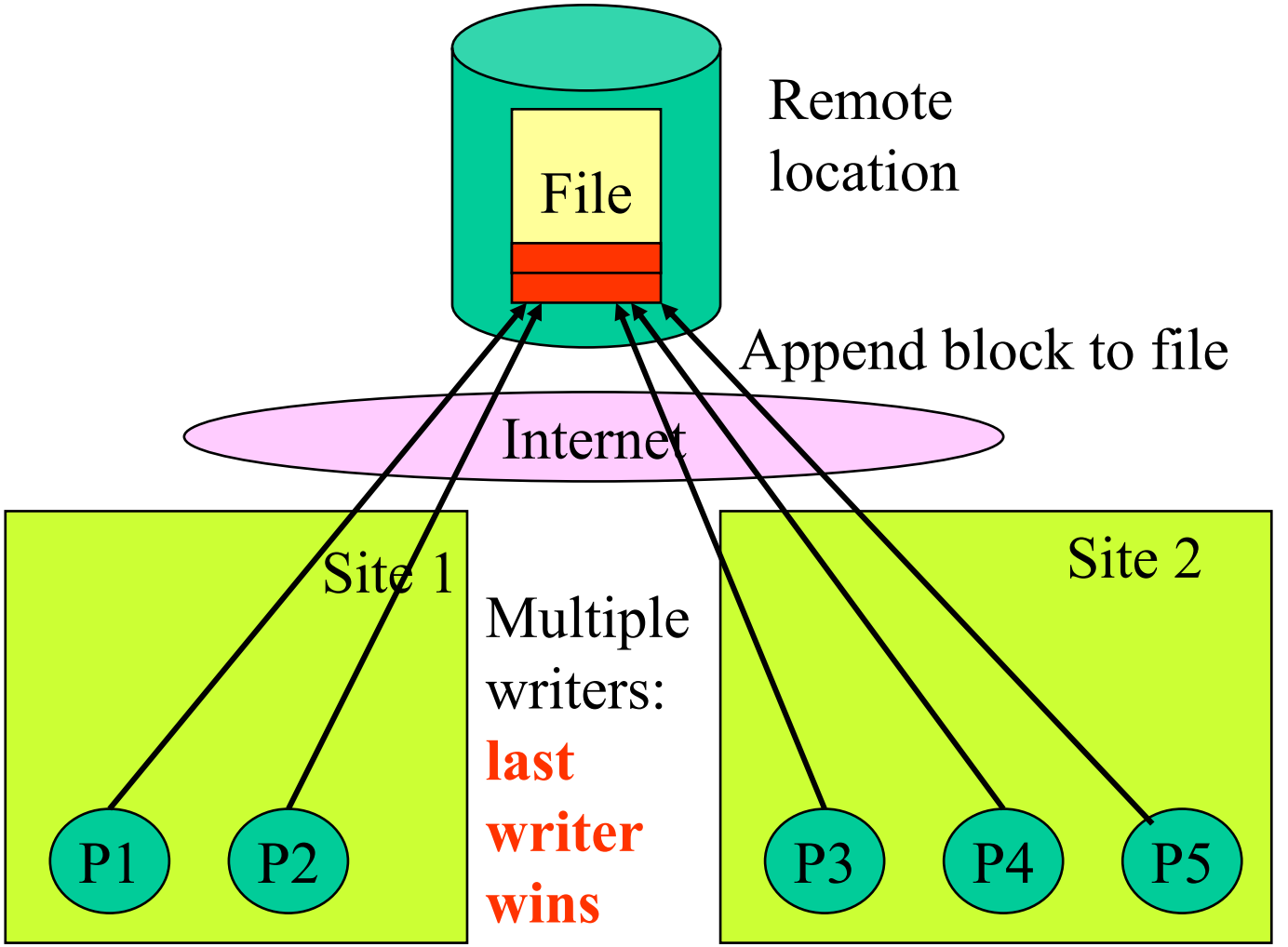
GASS File Close

- Solution: The reference count is checked.
  - If it is one, the file is copied back to the remote location and deleted from the cache
  - Otherwise, the reference count is decremented
- Advantage:
  - Reduces bandwidth requirements when multiple processes at the same location write to the same file
  - The file in cache can be accessed locally by conventional I/O calls
  - Conflicts are resolved locally, not remotely, and the file is transferred only once

Special case: write-only append

- Append to remote file if write only append: e.g., for stdout and stderr
- A remote file that is opened in write-only append mode is *not placed* in the cache
- Rather:
  - a communication stream is created to the remote location
  - Write operations to the file are translated into communication operations on that stream

Append-only access



Remote Cache Management Utilities

- Remote management of caches, for
  - Prestaging/poststaging of files
  - Cache cleanup and management
- Support operations on local & remote caches
- Functionality encapsulated in a program: globus-gass-cache

GASS Cache Semantics

- For each "file" in the cache, we record
  - Local file name
  - URL (i.e., the remote location)
  - Reference count: a set of tagged references
- Tags associated with references allow clean up of cache, e.g. following failure
  - Tag is job_manager_contact (if file accessed via file access API) or programmer-specified
  - Commands allow "remove all refs with tag T"

globus-gass-cache Specification

globus-gass-cache op [-r resource] [-t tag] URL

- Where op is one of
  - add : add URL to cache with tag
  - delete : remove one reference of tag for URL
  - cleanup_tag : remove all refs of tag for URL
  - cleanup_url : remove specified URL from cache
  - list : list contents of cache
- URL is optional for cleanup_tag and list
- If resource not specified, default to local cache

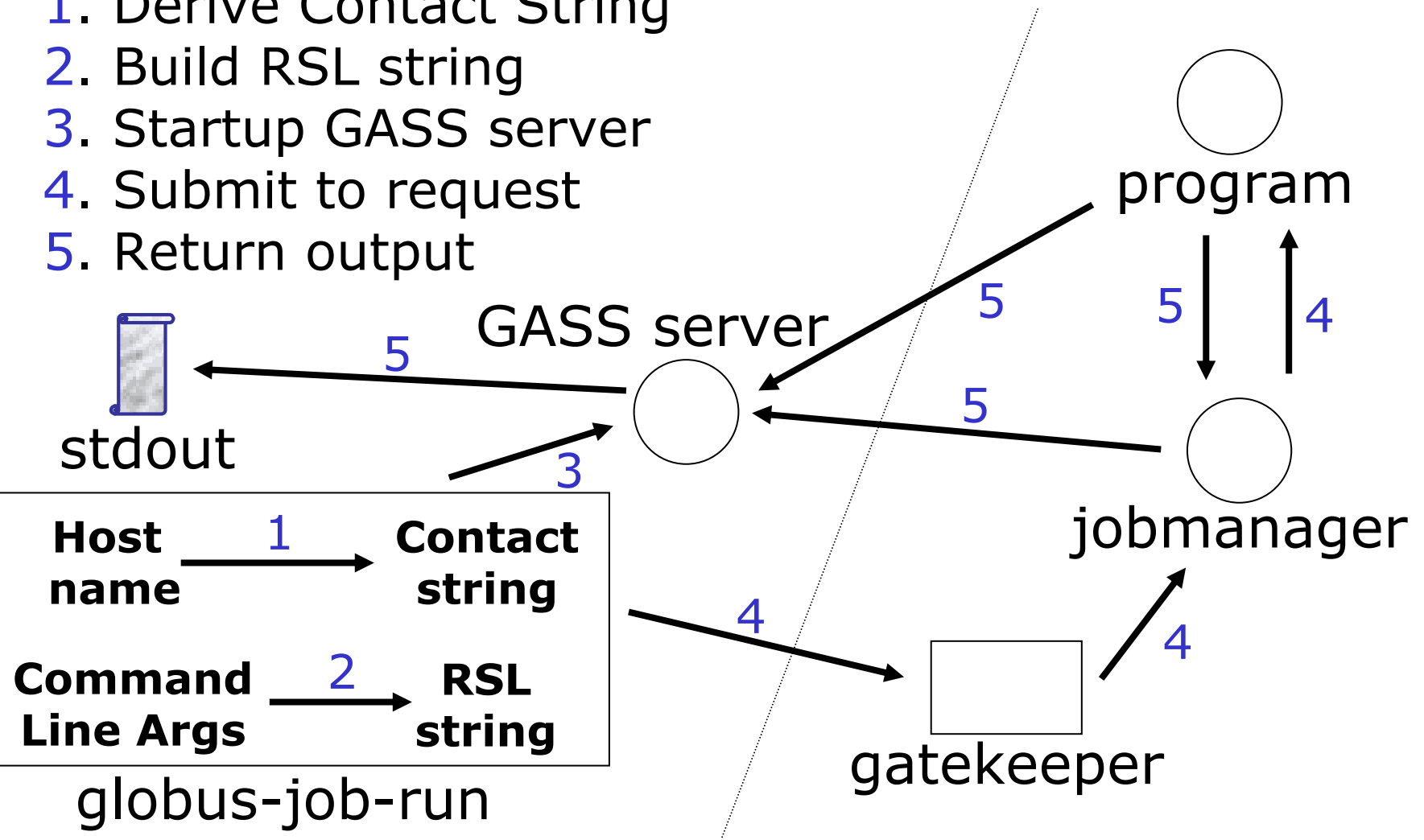
globus-gass-server

- Simple file server
  - Run by user wherever necessary
  - Secure https protocol, using GSI
  - APIs for embedding server into other programs
- Example

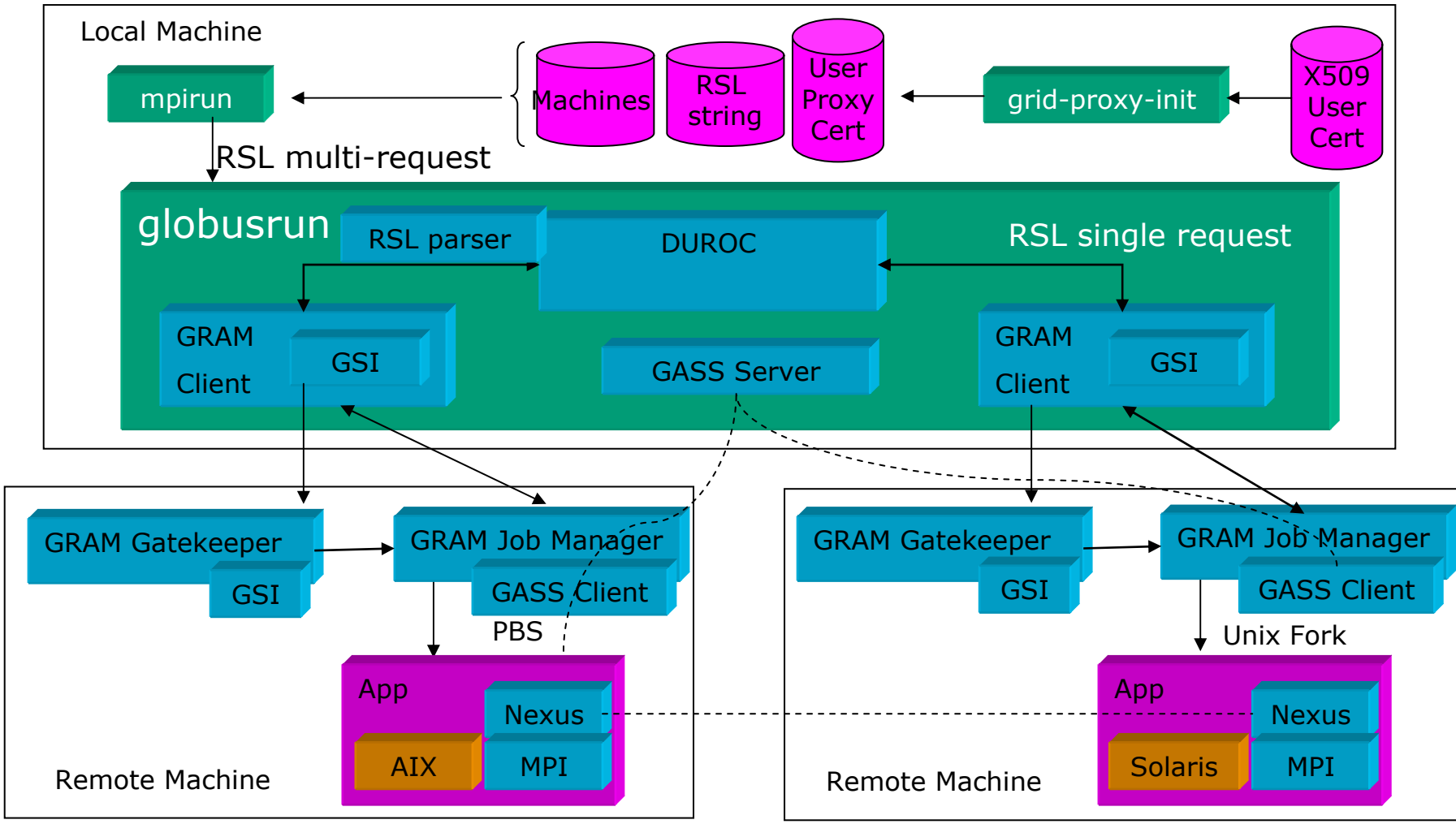
  globus-gass-server –r –w -t
  - -r: Allow files to be read from this server
  - -w: Allow files to be written to this server
  - -t: Tilde expand (~/… → $(HOME)/…)
  - -help: For list of all options

GRAM & GASS: Putting It Together

1. Derive Contact String
2. Build RSL string
3. Startup GASS server
4. Submit to request
5. Return output



Globus Components In Action



Köszönöm a figyelmüket

További információ: www.lpds.sztaki.hu