POOL File Catalog: Design & Status

Zhen Xie

CMS-Princeton/CERN

Outline

- Component overview
- Design
  - Requirements
  - Interface design
  - Use cases
  - Database schema
- Status
  - Infrastructure
  - Implementation
  - Performance test (update by Maria)

Component Overview

- Goal: maintain mapping between FileID (e.G. GUID) and PFNs for file based persistency;
- Text/XML,MySQL,EDG-RLS based implementations are forseen for the on/off the grid production at different scales;
- A very simple design to start with. First internal release by the end of September.

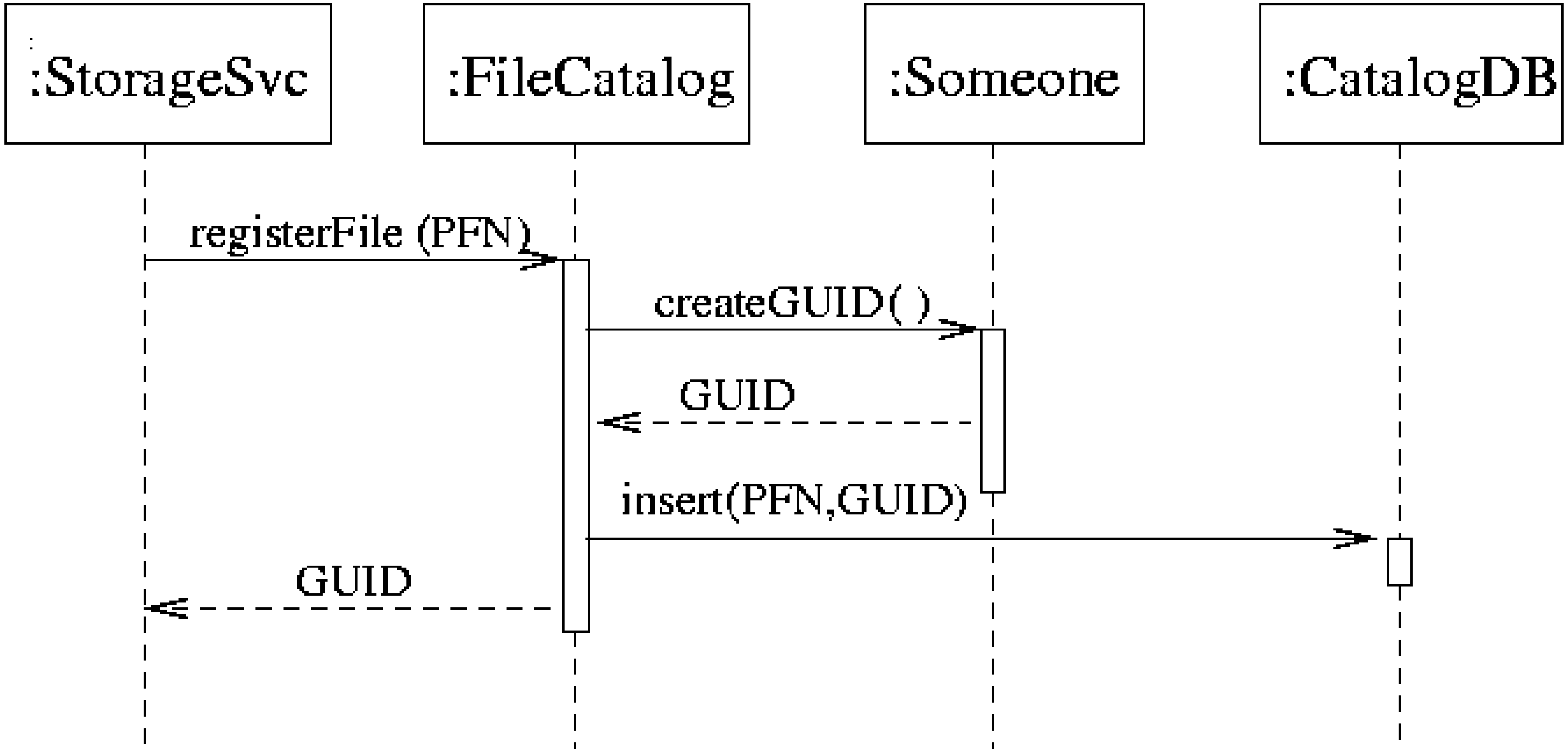
Requirements

- Unique and immutable FileID, memorisable text string LFN ;
- Provide PFN-FileID, LFN-PFN mappings;
- Register files before, after or in the job;
- Available on and off the grid;
- Can append to the same file from different jobs;
- Can clean up registration when job crashes.

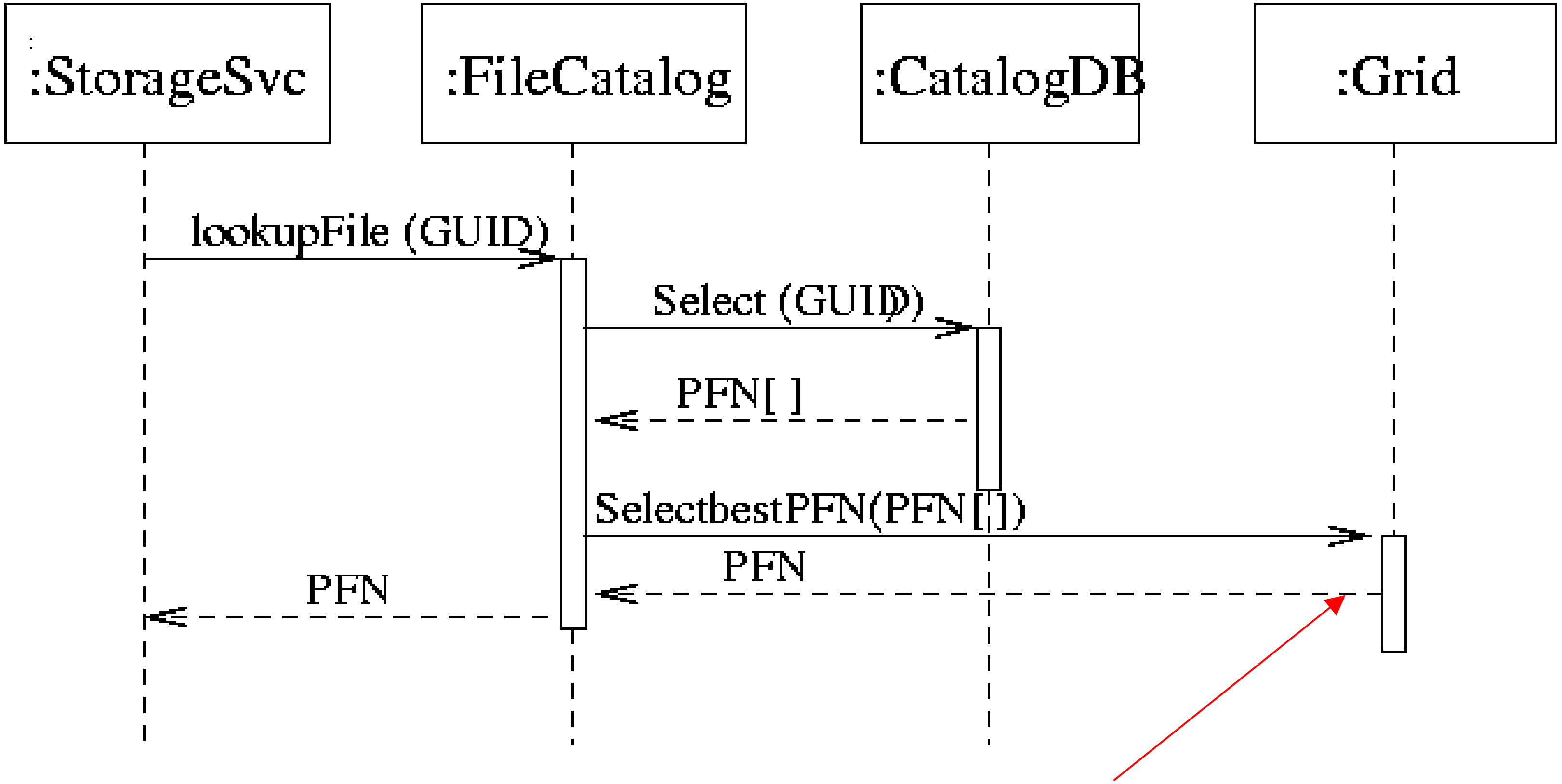
Design

- One common public interface IFileCatalog for two clients:
  - POOL storage manager
  - Command line tools for user
- Three implementations for three scopes of usage
  - Trivial text/XML catalog : local laptop user
  - Native MySQL catalog : production farm
  - EDG-RLS catalog : virtual organization
- Insulated interface and implementation

Use Case 1: Register File



Use Case 2: Lookup File



Off the Grid: the first PFN is returned

Use Case 3: Isolated Production

- Production manager pre-register PFNs into MySQL catalog;
- Production runs and the catalog is populated;
- Production ends. Production manager cleans up the catalog deleting file registration by unsuccessful jobs;
- Production manager assigns LFNs to file;
- Production manager publishes MySQL catalog fragment to the grid based catalog.

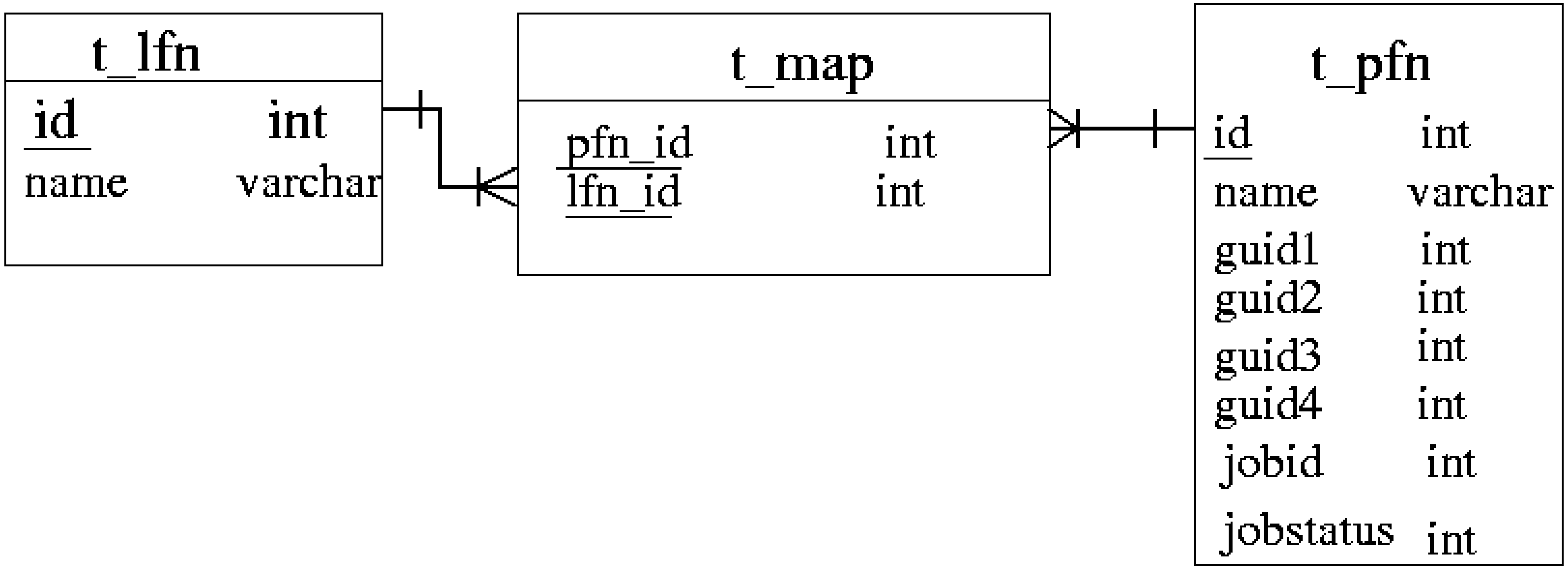
Use Case 4: Jobs on Disconnected Laptop

- User extracts the catalog fragmentation to be used in the job from a grid based catalog into a local text/XML catalog;
- One job starts and ends successfully. Output PFNs are registered into another text/XML catalog;

N jobs

- User publish selected text/XML catalogs into the grid based catalog.

Database Schema



t_lfn

id	int
name	varchar

t_map

pfn_id	int
lfn_id	int

t_pfn

id	int
name	varchar
guid1	int
guid2	int
guid3	int
guid4	int
jobid	int
jobstatus	int

Status: Infrastructure

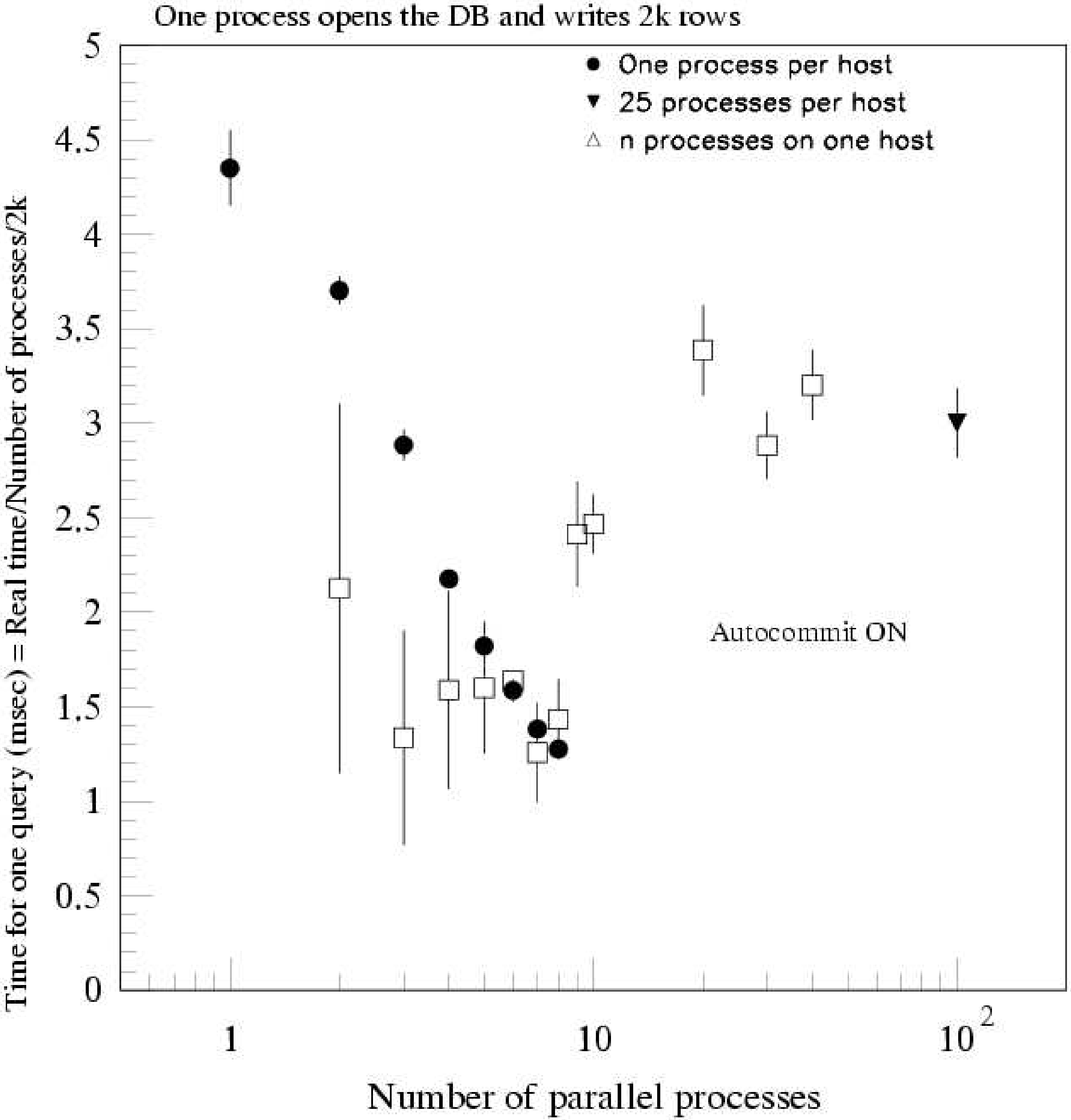
- cvs directories created
  - pool/FileCatalog
- External library
  - /afs/cern.Ch/sw/lcg/external/mysql++
- Component documentation: up-to-date
- Work package web site: up-to-date
  - http://lcgapp.cern.Ch/project/persist/catalog
- 1 MySQL server, 1 RLS server, 10 client nodes

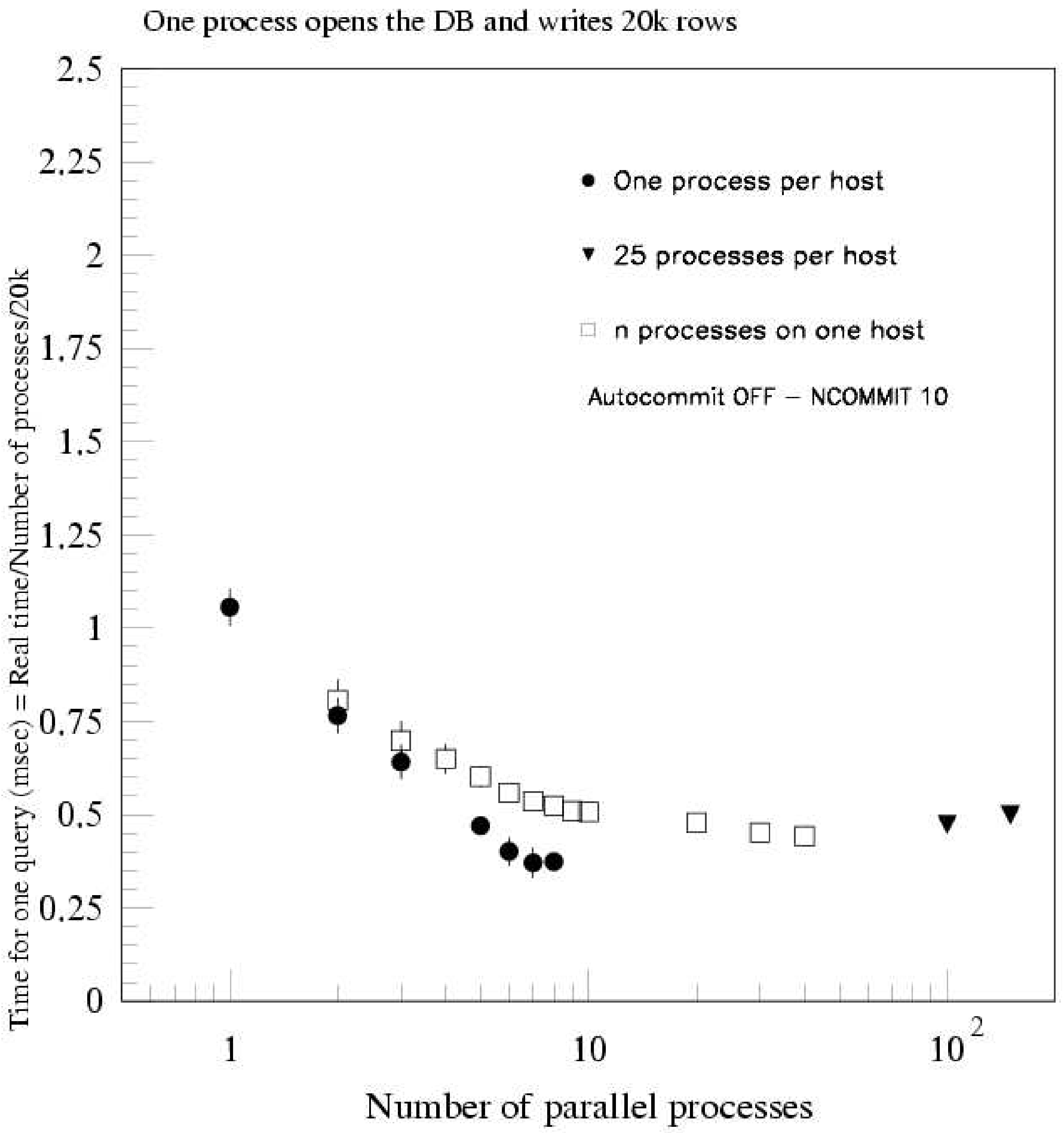
Status: Implementation

- Public interface
  - Pure virtual IFilecatalog, IFileCatalogConnection, IFileCatalogFactory
- MySQL catalog implementation
  - MySQLFileCatalog, MySQLFileCatalogConnection, MySQLFileCatalogFactory
  - MySQL catalog can connect to MySQL server

Status: Performance Test (Updated by Maria Girone)

- Native MySQL catalog performance.
  - Single process from single client via network: ~4ms/insert (autocommit on) see fig.1.
  - Concurrent access:
    - ~1.5 ms/insert (autocommit on) see fig.1.
    - <1 ms/insert (autocommit off) see fig.2.
- EDG-RLS catalog performance.
  - Always on autocommit mode.
  - Single process from single client via network: ~2ms/insert.
  - Concurrent access: server crash.


One process opens the DB and writes 2k rows
● One process per host
▼ 25 processes per host
△ n processes on one host
Autocommit ON
Time for one query (msec) = Real time/Number of processes/2k
Number of parallel processes
0
0,5
1
1,5
2
2,5
3
3,5
4
4,5
5
1
10
10^2



One process opens the DB and writes 20k rows
2,5
2,25
2
1,75
1,5
1,25
1
0,75
0,5
0,25
0
Time for one query (msec) = Real time/Number of processes/20k
● One process per host
▼ 25 processes per host
□ n processes on one host
Autocommit OFF – NCOMMIT 10
1
10
10 2
Number of parallel processes