

# The HPSS 5.1 Core Server

David Fisher, LLNL  
HPSS User Forum  
June 18, 2002

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

# Why a Core Server?

- ◆ Wanted to remove dependence on SFS
- ◆ Concerned that SFS might become unsupported
- ◆ Wanted to consider moving metadata to a database because DB systems are much more sophisticated now than 10 years ago

# Why a Core Server?

- ◆ We conducted research and now know databases can do the job
  - DB systems are well supported and powerful.
  - Our tables aren't large by DB standards.
  - Our data can be represented by DB types efficiently.
  - Transactions are central to DB design.

# Why a Core Server?

## ◆ DBs can do the job

### ■ Leverage

- ◆ Wide variety of management tools
  - Performance analysis
  - Backup
  - Handy keyboard interface
- ◆ Skilled people
- ◆ Wealth of information – books – training – user groups – expert consultants

# Why a Core Server?

## ◆ DBs can do the job

- Big gains in flexibility of handling HPSS metadata
  - ◆ New indices easily formed
  - ◆ Data can be easily reorganized
  - ◆ Ad hoc queries are easy

# Why a Core Server?

- ◆ If we're going to use a DB, HPSS should be designed like a DB application – a family of processes running transactions against database tables.

# Why a Core Server?

- ◆ But, the Name Server, Bitfile Server and Storage Servers use distributed transactions to perform basic HPSS functions.

# Why a Core Server?

- ◆ To continue using distributed transactions with a DB, we would have to use a transaction monitor such as Encina.

# Why a Core Server?

- ◆ But we want to reduce the number and cost of infrastructure components, so we must stop using distributed transactions.

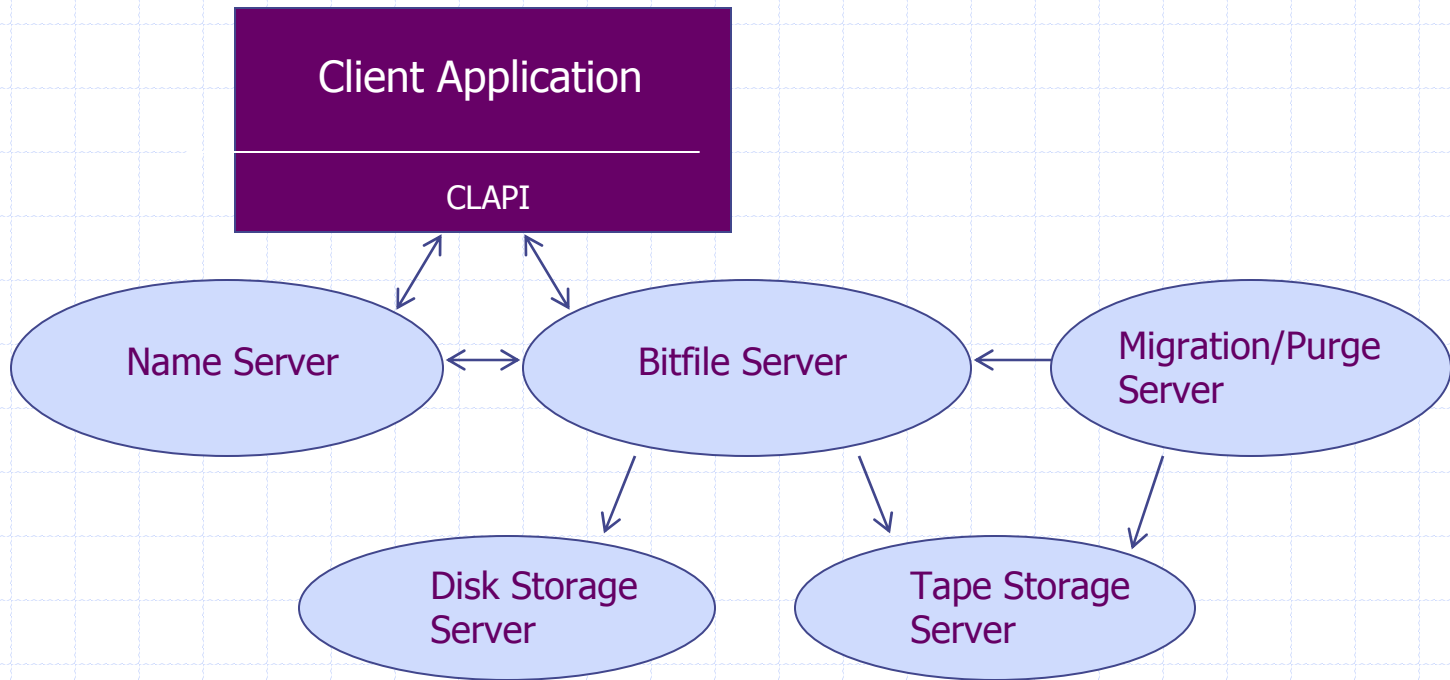
# Why a Core Server?

◆ Therefore create a single server program from:

- Client API components
- NS components
- BFS components
- SS components

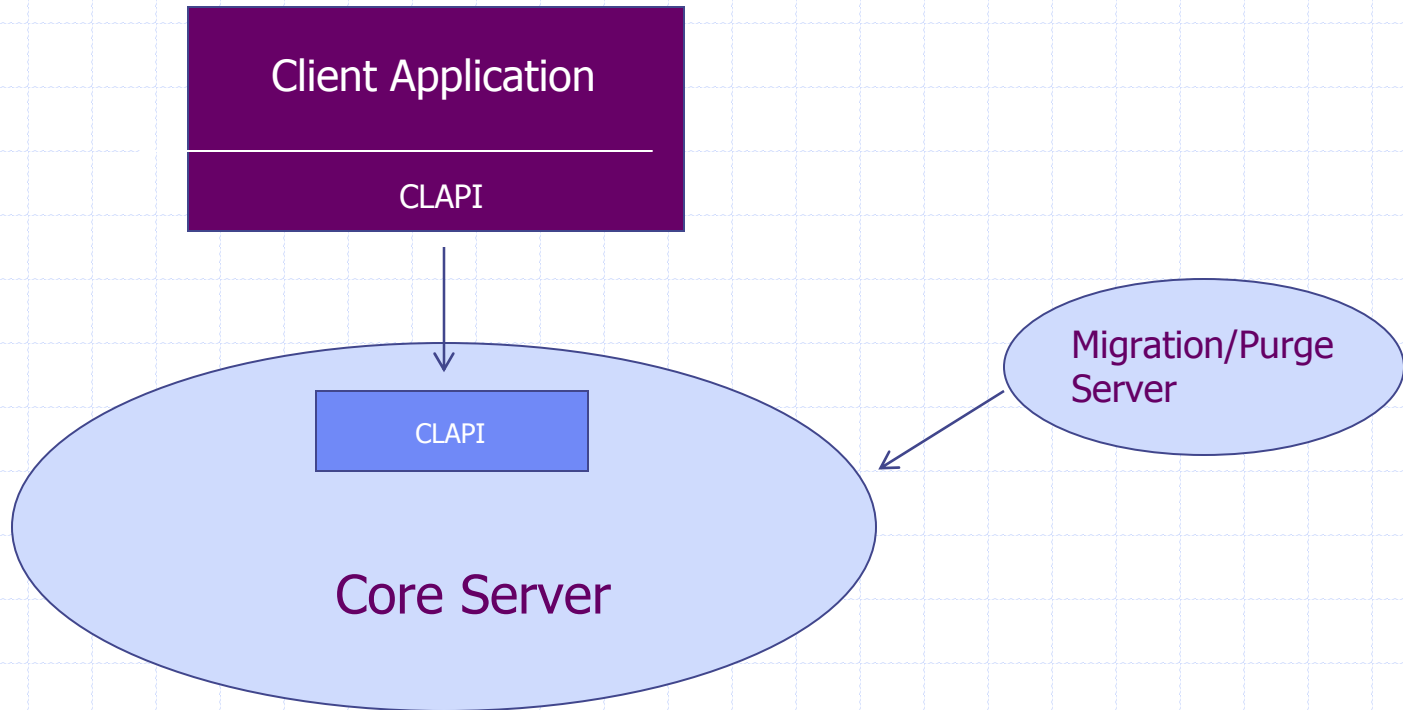
and eliminate the distributed transactions.

# What is the Core Server?



HPSS 4.5 Subsystem servers

# What is the Core Server?



HPSS 5.1 Subsystem servers

# What is the Core Server?

- ◆ Unifying NS, BFS and SS reduces the need for distributed transactions to almost zero.
  - NS has operations, used occasionally, that modify tables in both Global and SubSys DBs using a federated transaction feature.

# What is the Core Server?

- ◆ Unifying NS, BFS and SS increases performance
  - In 4.5 and earlier, many RPCs pass between NS, BFS and SS for each user request.
  - Transactions are distributed, take time to manage.
  - In the Core Server, function calls replace RPCs, and transactions are internal.

# What is the Core Server?

- ◆ Unifying NS, BFS and SS increases performance
  - Duplication of effort has been eliminated
    - ◆ Data used by two or more components can be shared instead of passed by RPC.
    - ◆ Components share common logging, RTM, signal handing, SSM driver, initialization and shutdown sub-systems.

# What is the Core Server?

## ◆ Architecture

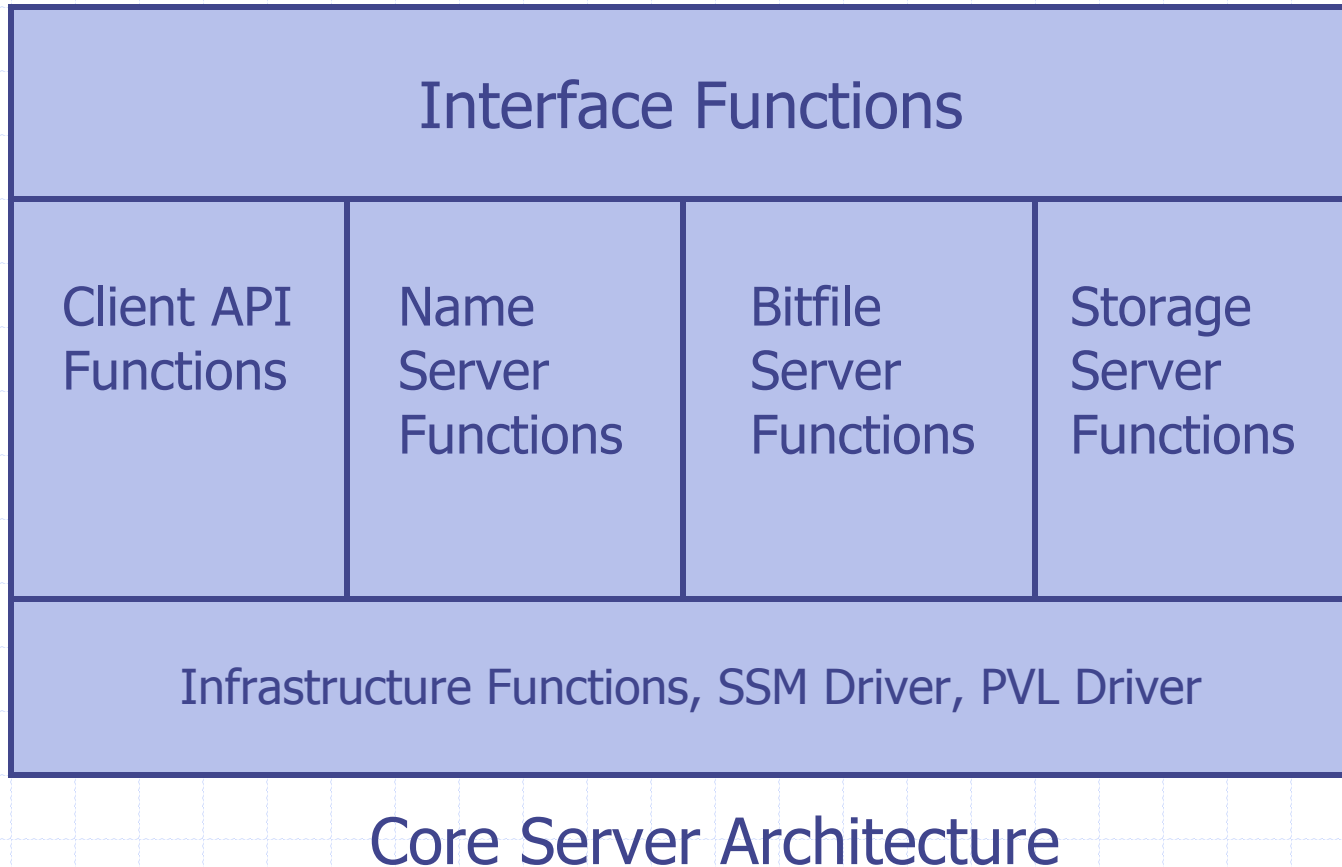
- New CS API functions
- Client API functions
- Name Server functions
- Bitfile Server functions
- Disk and Tape Storage Server functions

# What is the Core Server?

## ◆ Architecture

- Infrastructure components
  - ◆ Initialization, shutdown
  - ◆ Signal handling
  - ◆ Logging
  - ◆ Connection management
  - ◆ SSM driver
  - ◆ PVL driver

# How does it work?



# How does it work?

## ◆ DB configuration and MMLIB

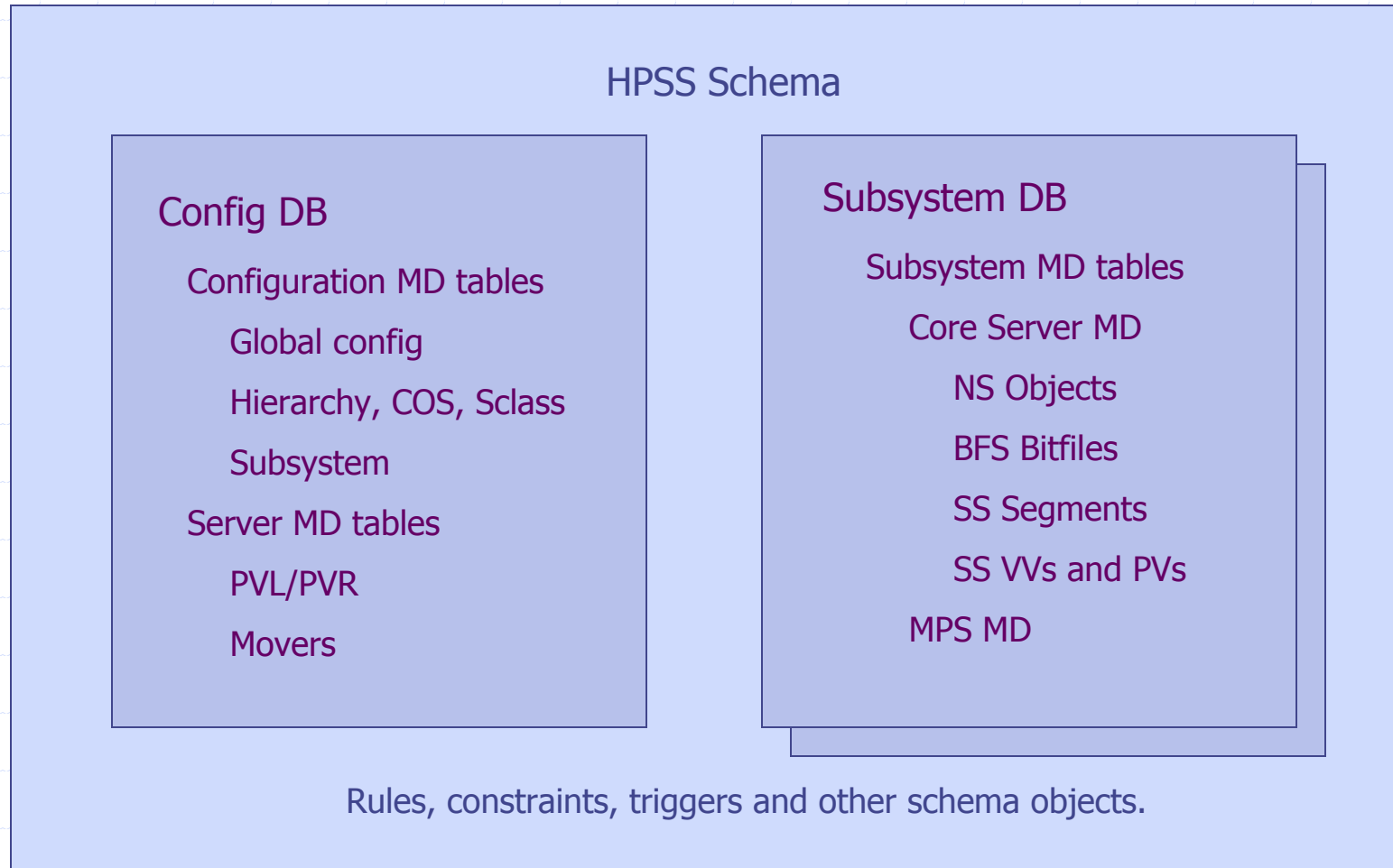
- Metadata is stored in conventional DB tables
- Core Server tables include
  - ◆ NS objects and ACLs
  - ◆ BFS bitfiles, disk and tape segments
  - ◆ SS maps, segments, VVs and PVs
- One database contains all the configuration tables and non-subsystem tables
- Another contains all the tables for a subsystem including CS tables

# How does it work?

## ◆ DB configuration and MMLIB

- Databases have variable names, table names are fixed
  - ◆ For any table, all you need to know to find MD is the DB Name and the Schema Name
- MMLIB provides create, delete, read and update functions for each table type

# How does it work?



# How does it work?

## ◆ Transaction management

- MMLIB provides start, commit, rollback and callback transaction services.
- MMLIB provides auto-commit and manual commit transactions.
- Core Server uses a set of macros that emulate Tran-C transaction control structures.

# How does it work?

## ◆ Transaction management

- No sub-transactions
- Transactions are started at the beginning of an API function and passed to the components.
- Metadata is modified as needed.
- Transaction commits at end of function.
- Most errors cause a rollback.

# What is the Core Server API?

- ◆ Functions in the API were contributed by CLAPI, NS, BFS and SSs.
- ◆ Functions made obsolete by CS metadata design are gone.
- ◆ Functions used only by CS components are no longer in the interface.

# What is the Core Server API?

## ◆ What's left?

- 67 functions!
- Most are NS and BFS functions, with some appropriate differences.
- Portions of CLAPI functions that run transactions.
- Some of the Disk and Tape SS functions.
- Some new functions.

# What is the Core Server API?

## ◆ How is the API different?

- Functions were re-designed to be uniform so that they look like HPSS had a CS from the beginning.
- Functions that had been transactional are now handled by larger meta-functions:
  - ◆ Core\_CreateStorageResource
  - ◆ Core\_DestroyStorageResource
  - ◆ Core\_CreateFile
- Other transactional functions – set attributes – have been made non-transactional at the interface.

# What is the Admin View?

- ◆ One Core Server per Storage Subsystem
- ◆ No NS, BFS, disk or tape SS in sight!
  - All of the generic and specific config records for these servers are gone.
- ◆ Each CS has a generic config record
  - Descriptive name
  - Server type
  - Storage subsystem ID

# What is the Admin View?

- ◆ Each CS has a specific config record
  - Contains a few server specific settings
  - But, there are no metadata file names to set
    - ◆ Metadata tables have fixed names wired into MMLIB
    - ◆ All you have to set is the Subsys database name.

# What is the Admin View?

- ◆ One Core Server managed object
  - Most statistics that used to be in NS, BFS and SS specific configs are now part of the CS managed object.
  - Global DB name
  - Subsystem DB name
  - Schema name

# What is the Admin View?

- ◆ One Core Server managed object
  - Bitfile limits – max open files, max IO requests, etc.
  - Root fileset name and ID
  - Name server object counts
  - Storage server volume and byte counts

# Current status

- ◆ CS is running
- ◆ Creates/deletes storage resources
- ◆ Creates/deletes files
- ◆ Reads/writes disk and tape
- ◆ System testing begins soon