# A Dynamic Prefix Tree for the Service Discovery Within Large Scale Grids

Eddy Caron, Frédéric Desprez, Cédric Tedeschi

GRAAL Project / LIP laboratory
UMR CNRS-ENS Lyon-UCB Lyon-INRIA 5668

P2P 2006 - September 6, 2006

# Outline

# Outline

# Context : service discovery in computational grids

- *Service*
  - binary file, library (High performance computing context)
  - pre-installed on servers
  - servers registers their services
  - clients wish to discover them
- Characteristics of emerging grids
  - large scale
  - no central infrastructure
  - dynamic joins and leaves of nodes
- Adopt peer-to-peer technologies
  - purely decentralized algorithms
  - scalable algorithms to retrieve objects
  - fault-tolerance

# Context : service discovery in computational grids

- *Service*
  - binary file, library (High performance computing context)
  - pre-installed on servers
  - servers registers their services
  - clients wish to discover them
- Characteristics of emerging grids
  - large scale
  - no central infrastructure
  - dynamic joins and leaves of nodes
- Adopt peer-to-peer technologies
  - purely decentralized algorithms
  - scalable algorithms to retrieve objects
  - fault-tolerance

# Context : service discovery in computational grids

- *Service*
  - binary file, library (High performance computing context)
  - pre-installed on servers
  - servers registers their services
  - clients wish to discover them
- Characteristics of emerging grids
  - large scale
  - no central infrastructure
  - dynamic joins and leaves of nodes
- Adopt peer-to-peer technologies
  - purely decentralized algorithms
  - scalable algorithms to retrieve objects
  - fault-tolerance

# Requirements

- Functionnalities required (registering and discovering)
  - Automatic completion/range queries
  - Multicriteria search
- Efficiency requirements
  - Purely decentralized system (P2P)
  - Fault tolerance
  - Physical locality awareness

# Requirements

- Functionnalities required (registering and discovering)
  - Automatic completion/range queries
  - Multicriteria search
- Efficiency requirements
  - Purely decentralized system (P2P)
  - Fault tolerance
  - Physical locality awareness

# Services description

- A service is described by a set of attributes :

**S** = { **name**, **OS**, **processor**, **location** }

S = { DGEMM, Linux Debian 3, PowerPC G5, com.grid.n1 }

**(key, value)** pairs

(DGEMM, n1.grid.com)

(Linux Debian 3, n1.grid.com)

(PowerPC G5, n1.grid.com)

(com.grid.n1, n1.grid.com)

# Outline

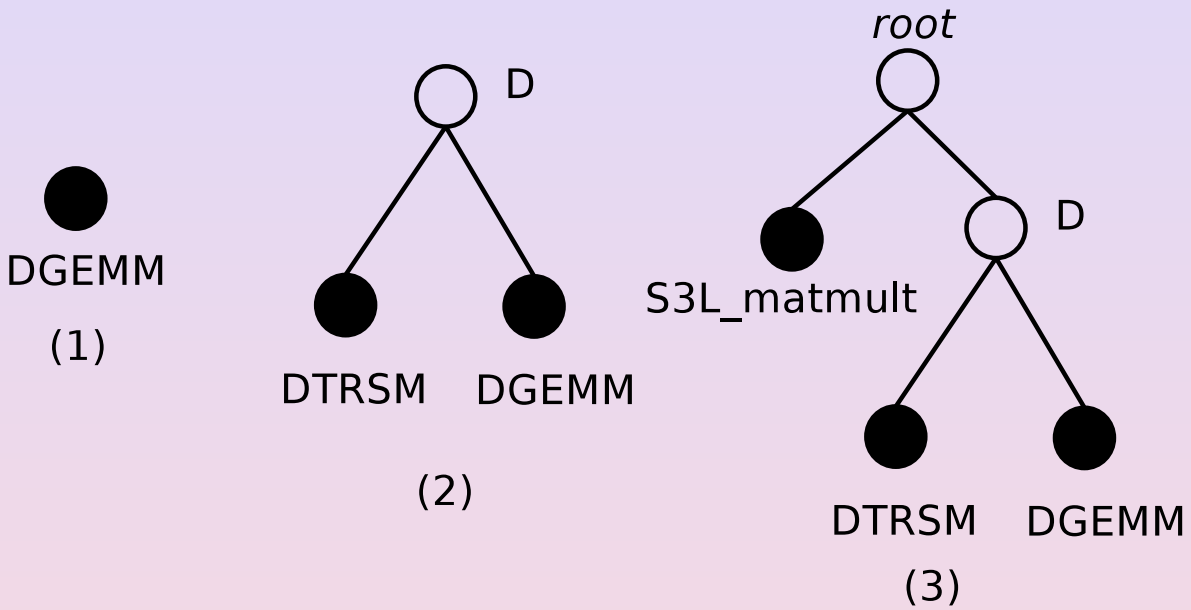
# General design

- DLPT : **D***istributed* **L***exicographic* **P***lacement* **T***able*
- A two layers architecture
  - Logical layer
    - On-line building of a Greatest Common Prefix Tree with attributes of services declared
    - Replication based fault-tolerance
    - Greedy locality awareness
  - Dynamic mapping of logical nodes on the physical nodes
    - DHT-based (load balancing)
    - a physical node hosts one or more logical nodes

# Logical structure - GCP Tree

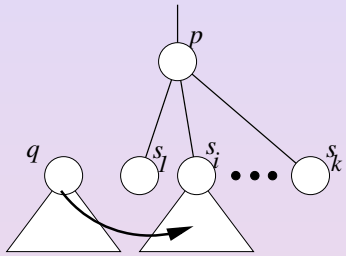
- Alphabet $A$ finite set of letters
- $\prec$ an order on $A$
- Word $w$ finite sequence of letters of $A$, $w = a_1, \ldots, a_i, \ldots, a_l$, $l > 0$
- $u, v$ two words, $uv$ *concatenation* of $u$ and $v$
- $|w|$ length of $w$
- $\epsilon$ the empty word, $|\epsilon| = 0$
- $u = prefix(v)$ if $\exists w$ s.t. $v = uw$
- $GCP(w_1, w_2, \ldots, w_i, \ldots, w_n)$ is the longest prefix shared by $w_1, w_2, \ldots, w_i, \ldots, w_n$
- A GCP Tree is a labeled rooted tree s.t.
  - The node label is a proper prefix of any label in its subtree
  - The node label is the Proper Greatest Common Prefix of all its son labels

# On-line construction

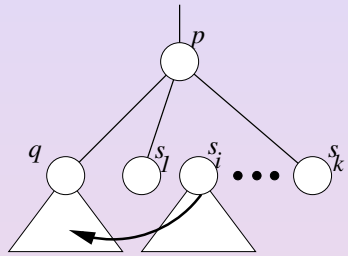



- Contact by a server who wants to register a service
- Routing of the request
- Inserting of the (key, value) pair

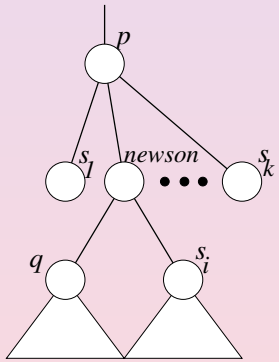
# Distributed routing in a GCP Tree



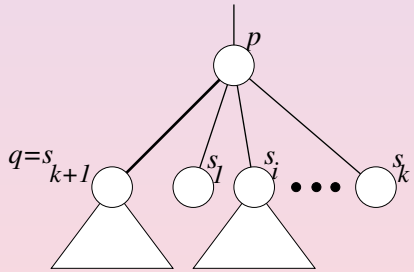

(*a*) There exists $s_i$ such that $s_i.val = prefix(q.val)$.




(*b*) There exists $s_i$ such that $q.val = prefix(s_i.val)$.




(*c*) There exists $s_i$ such that $GCP(q.val, s_i.val) > p.val$.




(*d*) $p.val = prefix(q.val)$.

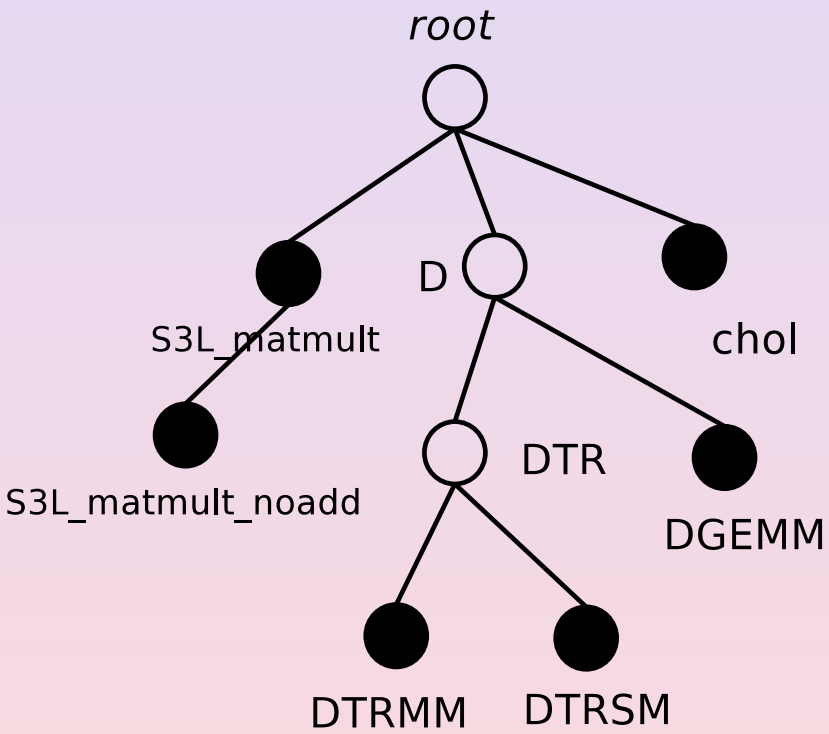
# Fault Tolerance and locality awareness (1/2)

- Static replication factor $k$
- Greedy locality awareness
- Periodically initiated by the root
- Replication of the root
- Election of one replica to launch the process in the subtree

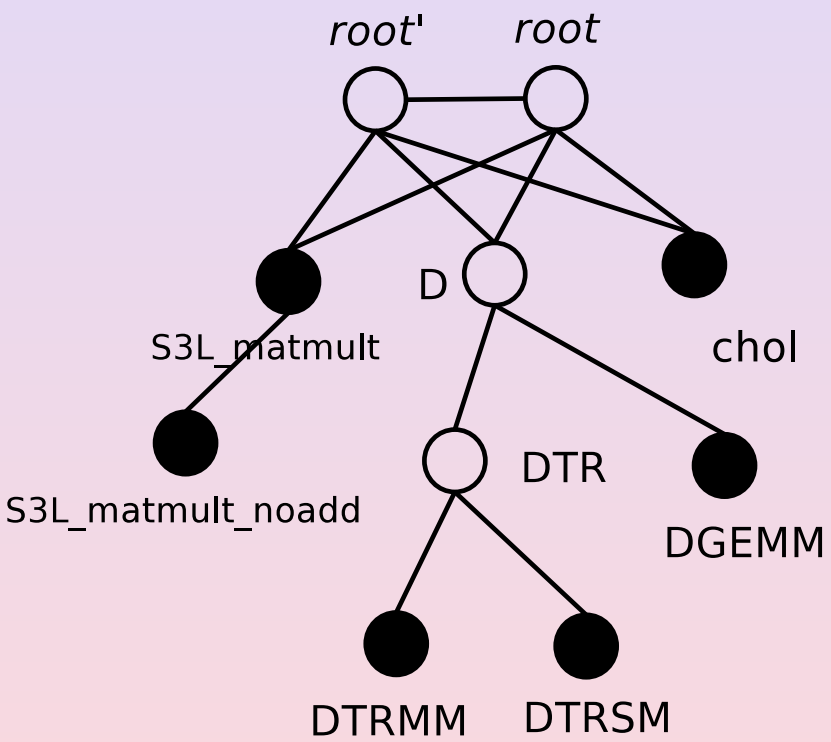
# Fault Tolerance and locality awareness (1/2)

- Static replication factor $k$
- Greedy locality awareness
- Periodically initiated by the root
- Replication of the root
- Election of one replica to launch the process in the subtree

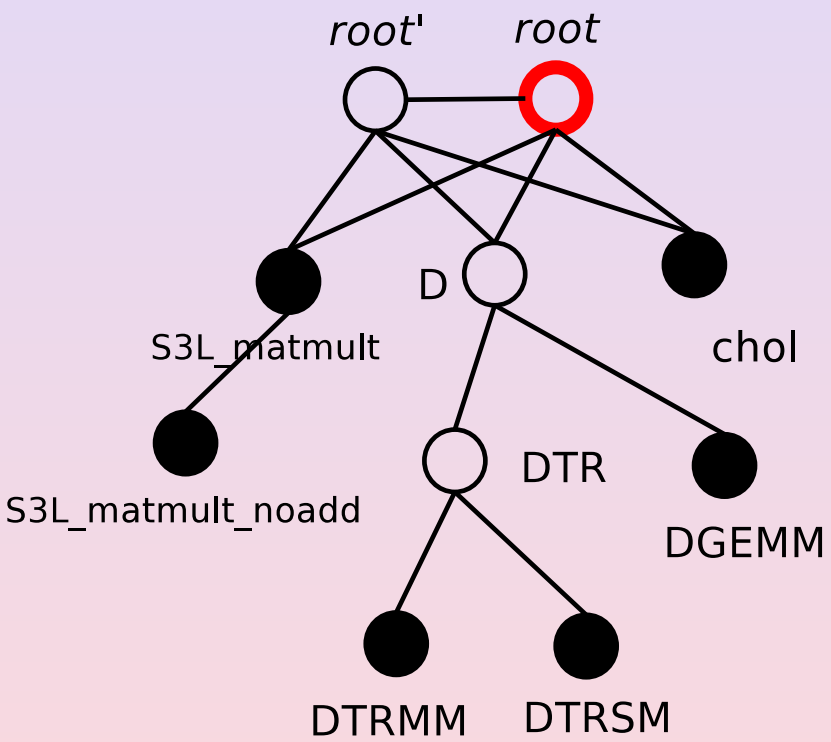
# Fault Tolerance and locality awareness (2/2)



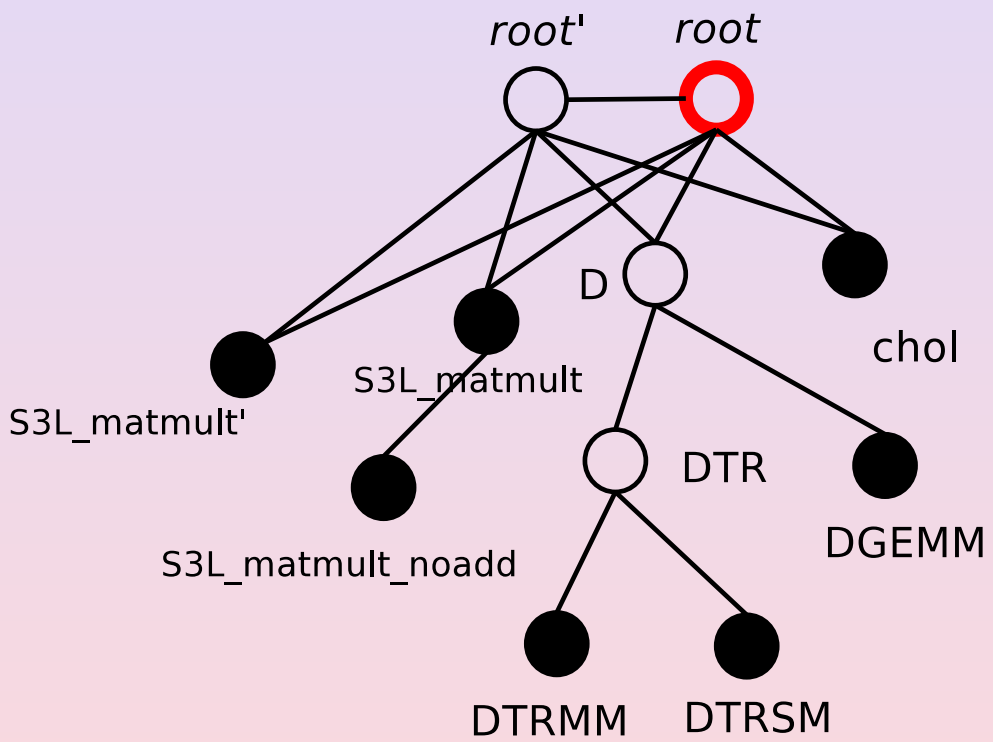
# Fault Tolerance and locality awareness (2/2)



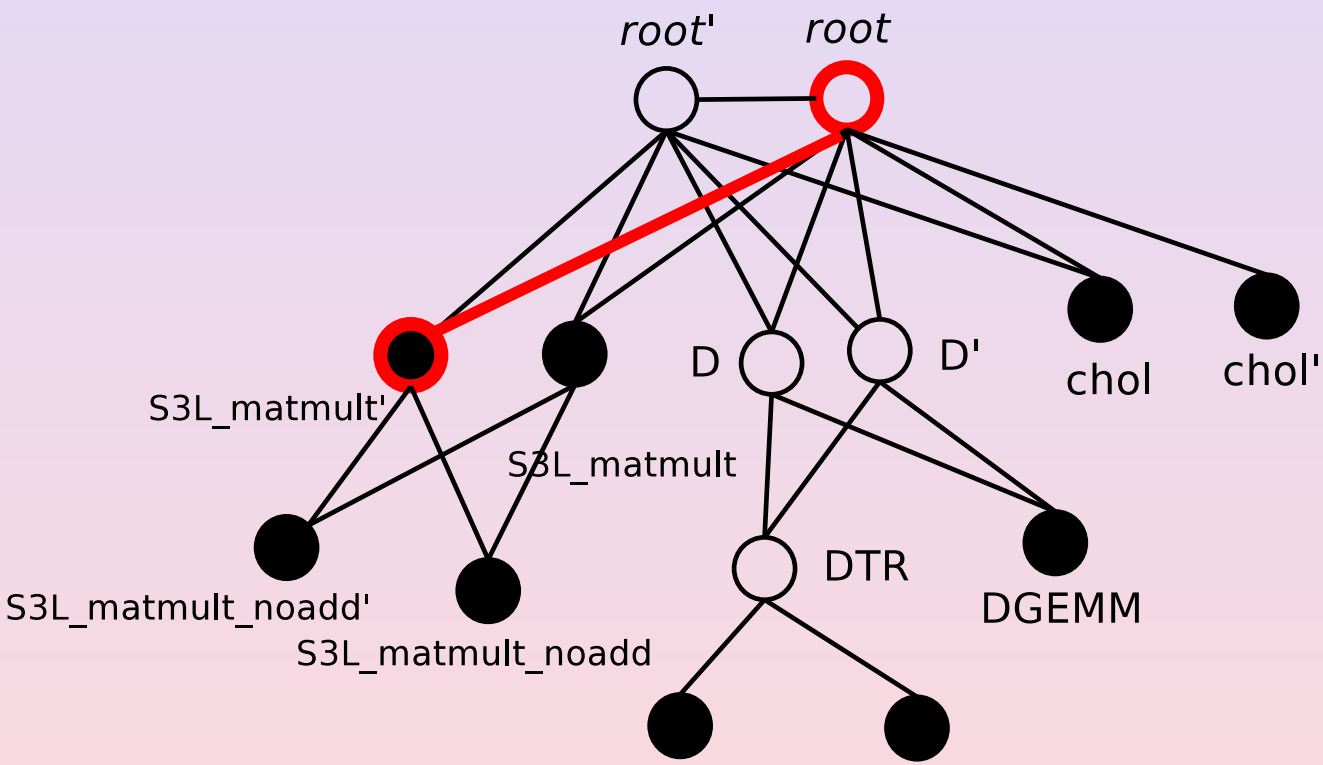
# Fault Tolerance and locality awareness (2/2)



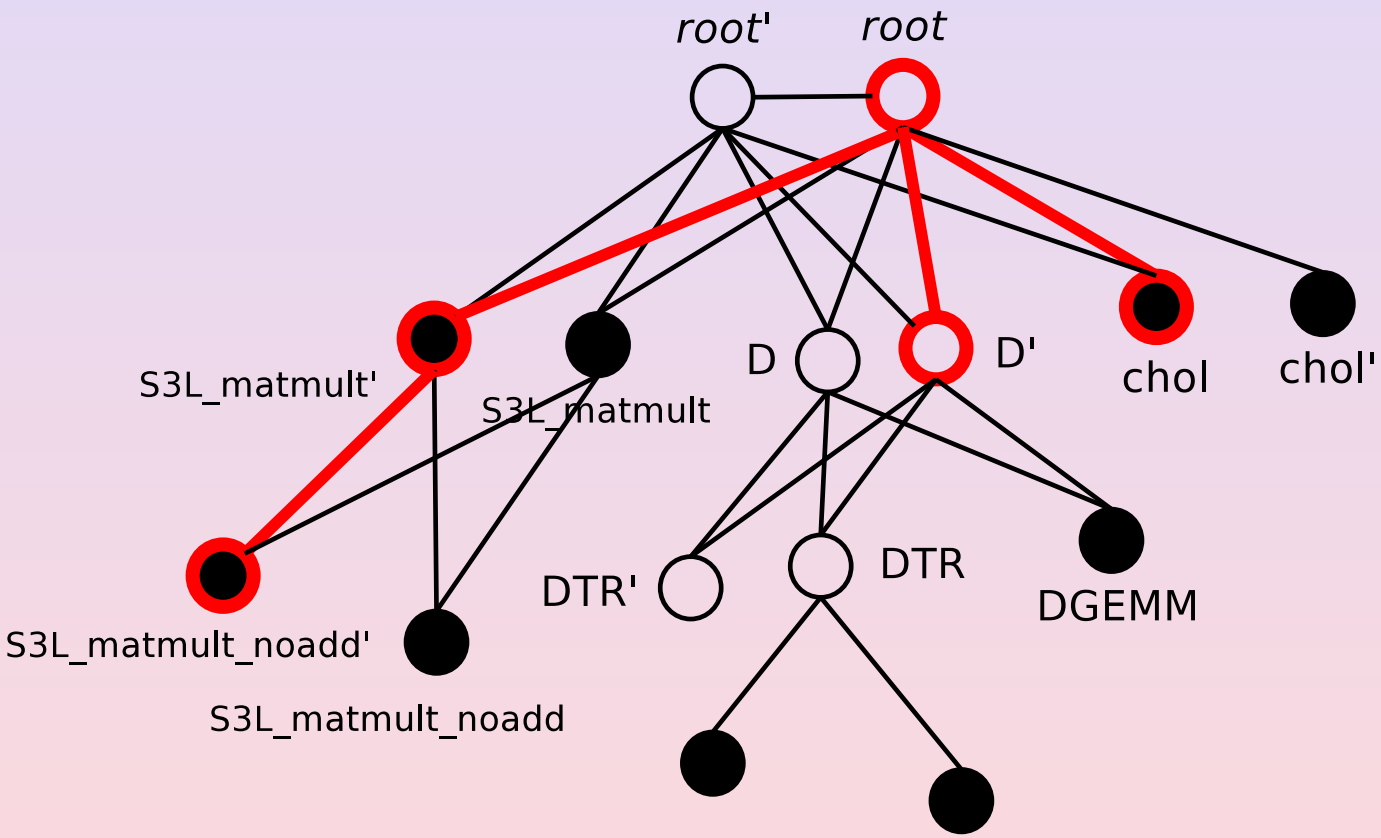
# Fault Tolerance and locality awareness (2/2)



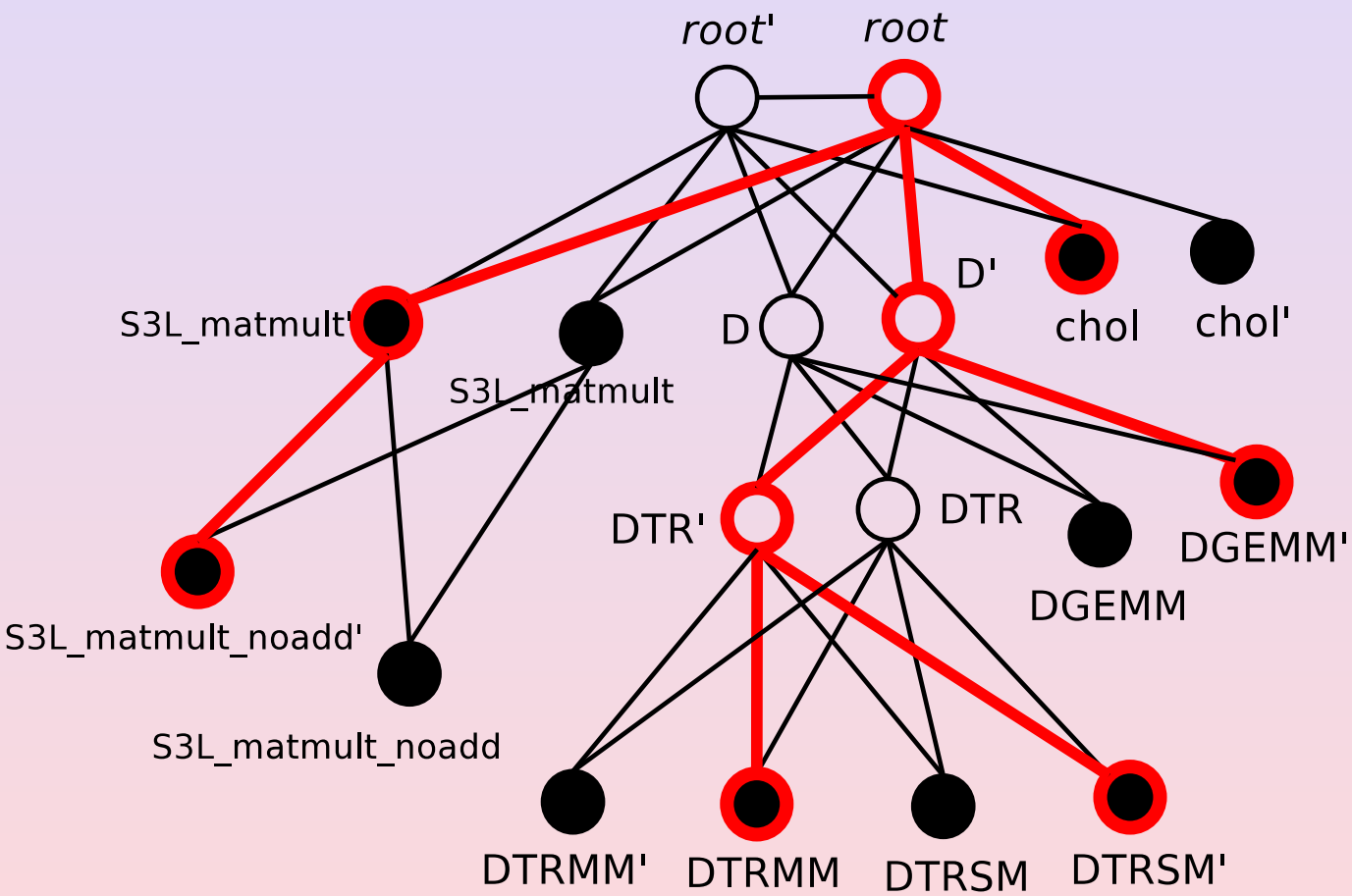
# Fault Tolerance and locality awareness (2/2)



# Fault Tolerance and locality awareness (2/2)

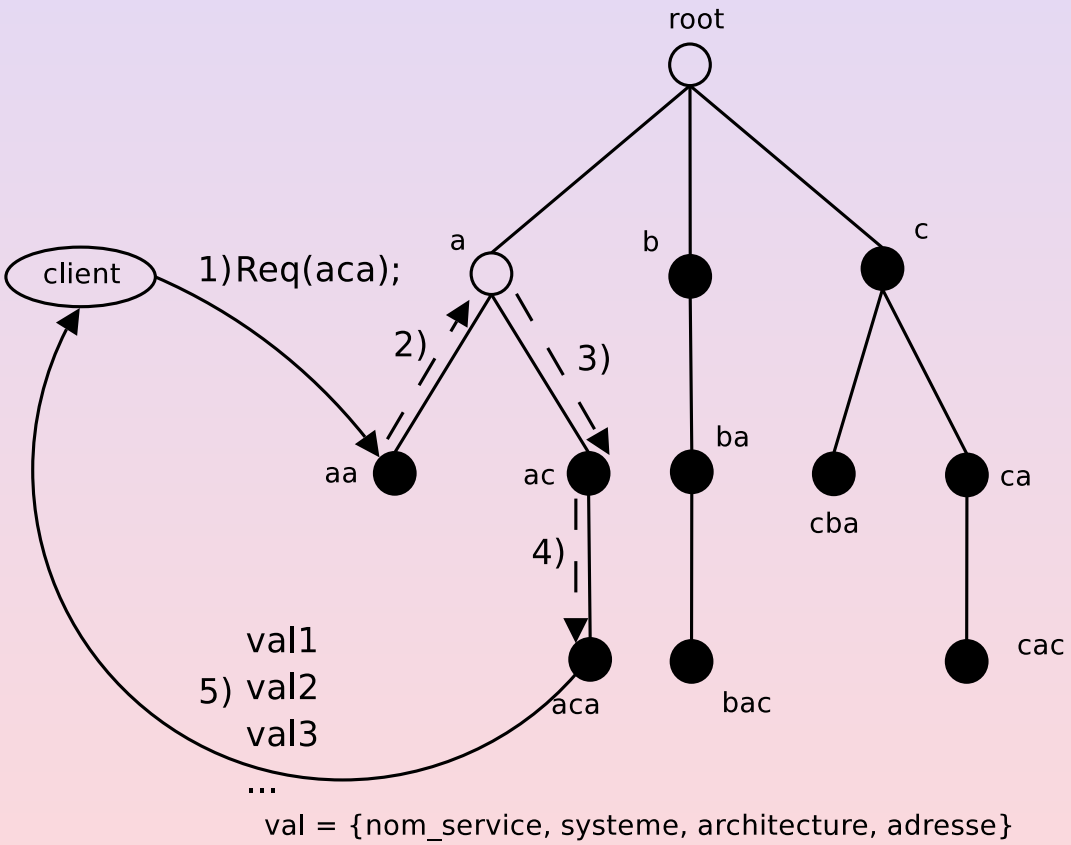


# Fault Tolerance and locality awareness (2/2)



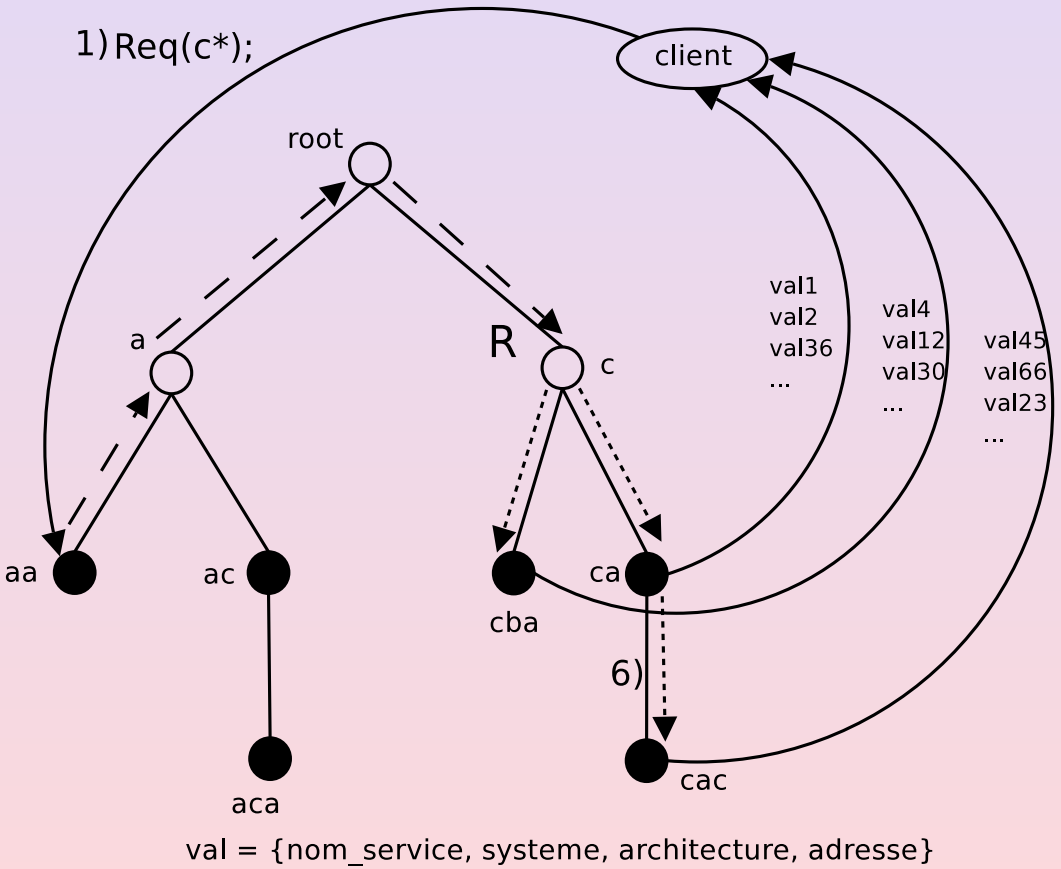
# Querying

- Exact match query
- Range query (automatic completion)
- Multicriteria lookup



# Querying

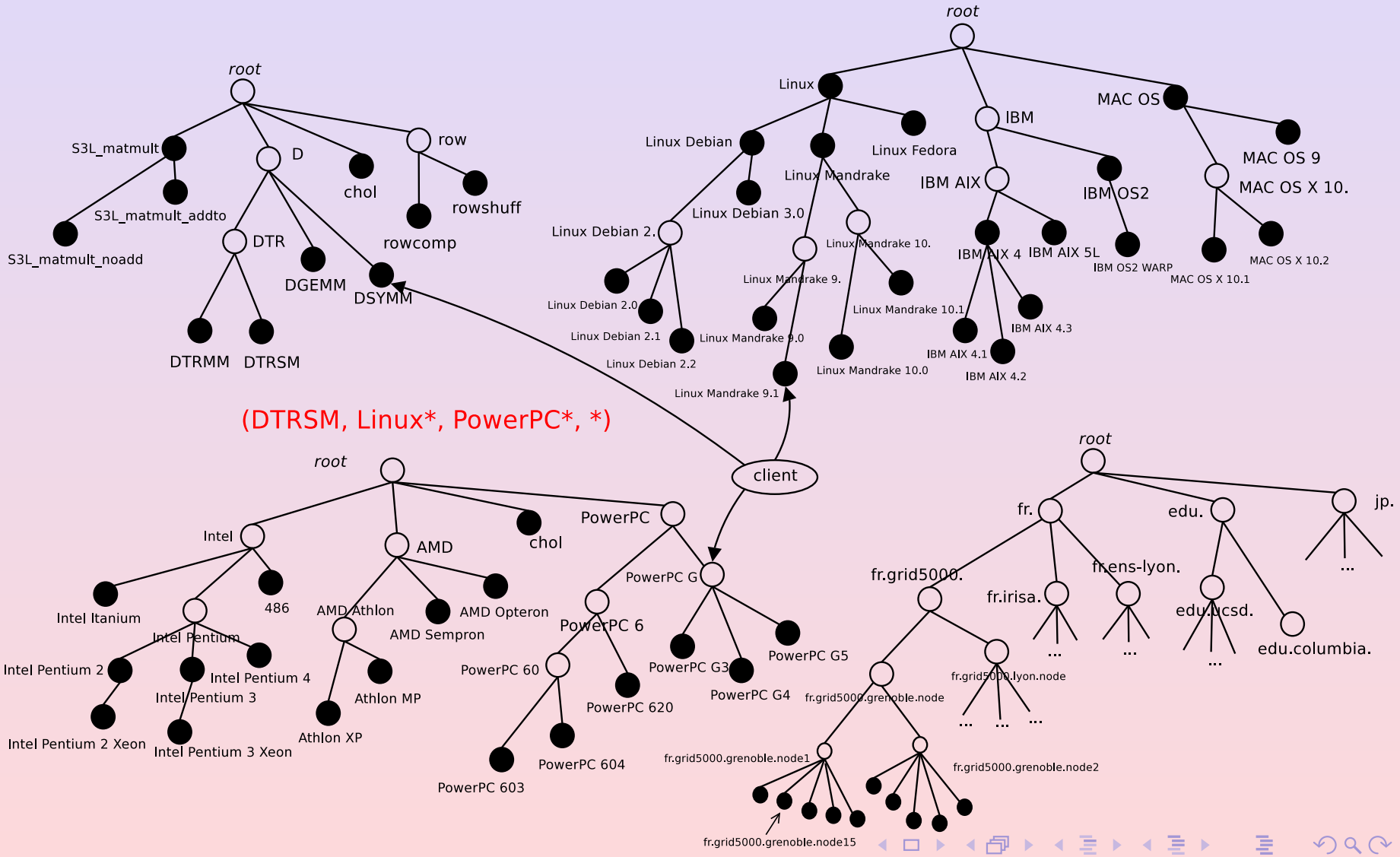
- Exact match query
- Range query (automatic completion)
- Multicriteria lookup



# Querying

- Exact match query
- Range query (automatic completion)
- Multicriteria lookup

# Querying


root
S3L_matmult
S3L_matmult_noadd
S3L_matmult_addto
D
DTR
DTRMM
DTRSM
DGEMM
DSYMM
chol
row
rowcomp
rowshuff
root
Linux
Linux Debian
Linux Debian 2.
Linux Debian 2.0
Linux Debian 2.1
Linux Debian 2.2
Linux Debian 3.0
Linux Mandrake
Linux Mandrake 9.
Linux Mandrake 9.0
Linux Mandrake 9.1
Linux Mandrake 10.
Linux Mandrake 10.0
Linux Mandrake 10.1
Linux Fedora
IBM
IBM AIX
IBM AIX 4
IBM AIX 4.1
IBM AIX 4.2
IBM AIX 4.3
IBM AIX 5L
IBM OS2
IBM OS2 WARP
MAC OS
MAC OS 9
MAC OS X 10.
MAC OS X 10.1
MAC OS X 10.2
(DTRSM, Linux*, PowerPC*, *)
client
root
Intel
Intel Itanium
Intel Pentium
Intel Pentium 2
Intel Pentium 2 Xeon
Intel Pentium 3
Intel Pentium 3 Xeon
Intel Pentium 4
486
AMD
AMD Athlon
AMD Sempron
AMD Opteron
Athlon MP
Athlon XP
chol
PowerPC
PowerPC G
PowerPC 6
PowerPC 60
PowerPC 603
PowerPC 604
PowerPC 620
PowerPC G3
PowerPC G4
PowerPC G5
root
fr.
fr.grid5000.
fr.grid5000.grenoble.node
fr.grid5000.grenoble.node1
fr.grid5000.grenoble.node15
fr.grid5000.grenoble.node2
fr.irisa.
fr.ens-lyon.
fr.grid5000.lyon.node
edu.
edu.ucsd.
edu.columbia.
jp.

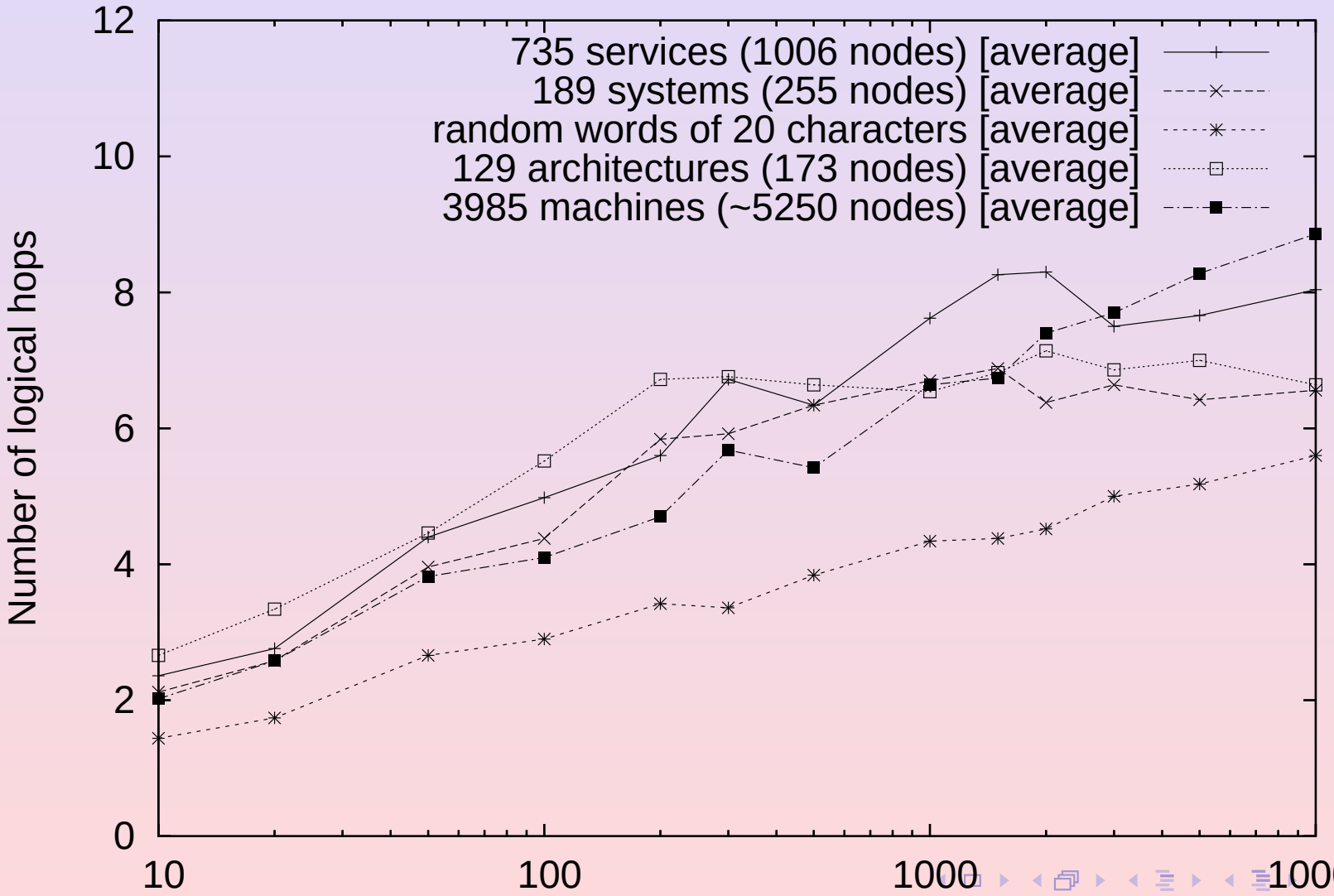
# Outline

# Complexities

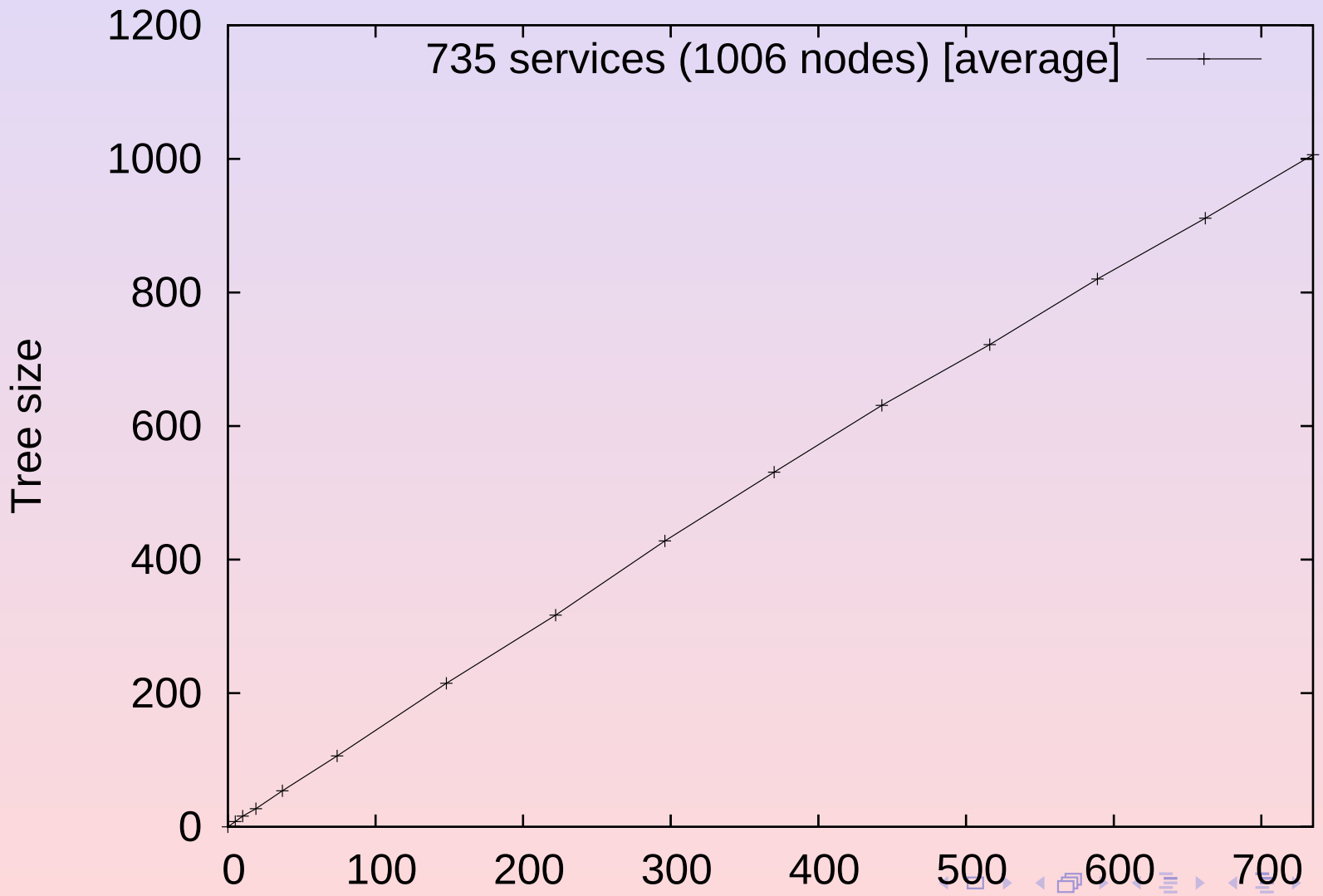
- Number of hops of routing bounded by twice the depth of the tree
- Local state bounded by the number of characters in the alphabet used
- Constant time local decision of routing
- Range query, replication/locality process
  - latency bounded by the depth of the tree
  - linear number of messages

# Outline

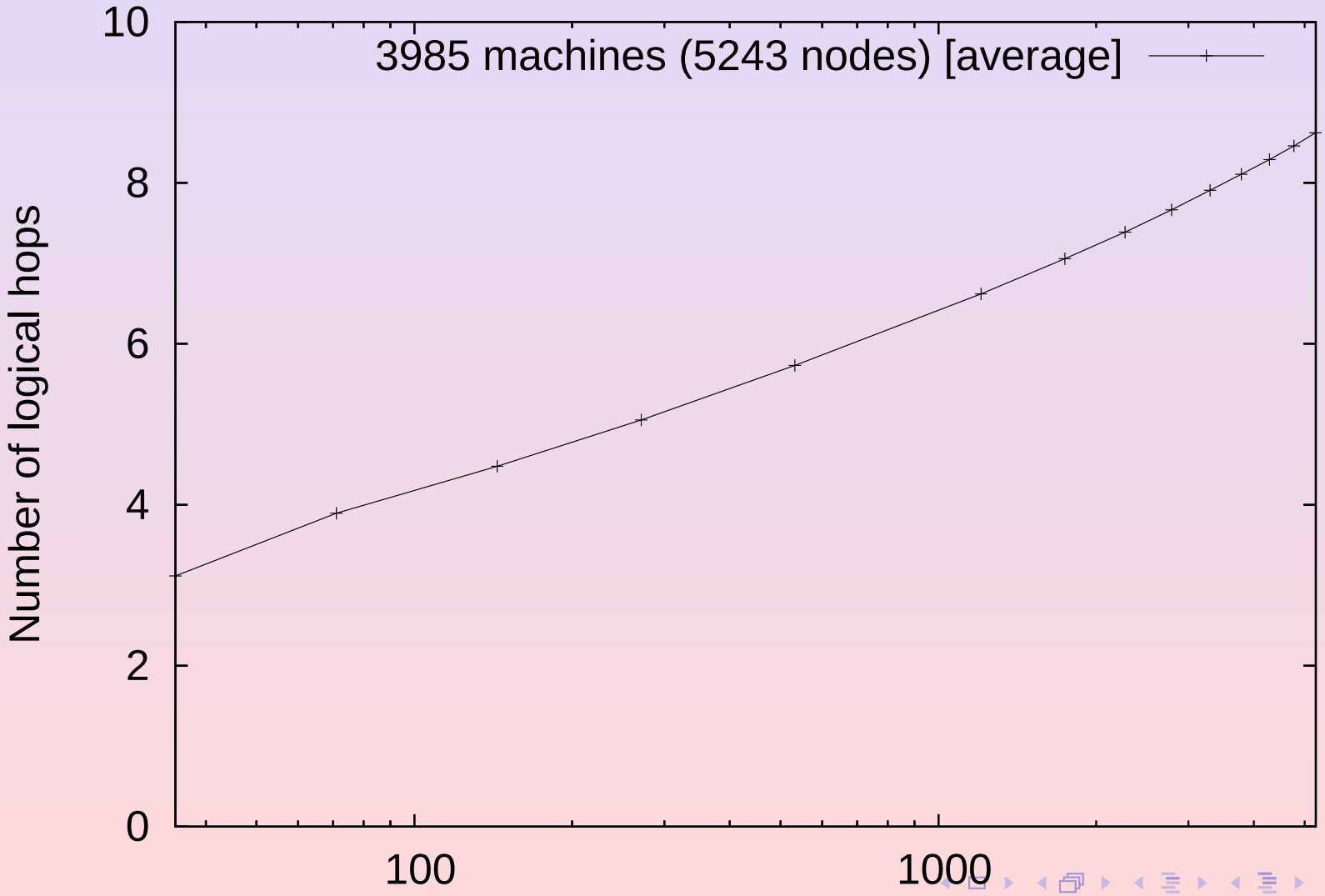
# Simulation - logical hops (insertion request)



# Simulation - size of the tree



# Simulation - logical hops (interrogation requests)



# Outline

# Conclusion

- Contribution : DLPT
  - distributed architecture for the service discovery in large scale grids
  - dynamic prefix tree
  - fault-tolerant (k-replication)
  - locality aware prefix tree
  - comparisons with similar approaches in the paper (Skip Graphs, Nodewiz, Prefix Hash Tree, P-Grid)
- Current and future work
  - repair mechanism avoiding complete replication
  - prototype and experiments on a real grid
  - theoretical study of an efficient mapping and load balancing (avoiding an underlying DHT)

# P2P technologies

- Unstructured P2P approaches
  - flooding based
  - non-exhaustive searches
- Distributed Hash Tables
  - routing based
  - exhaustive search
  - scalable :
    - logarithmic local state
    - logarithmic number of hops
  - fault-tolerance
    - periodic scanning
    - replication
  - drawbacks
    - only exact match queries
    - no locality awareness

# P2P technologies

- Unstructured P2P approaches
  - flooding based
  - non-exhaustive searches
- Distributed Hash Tables
  - routing based
  - exhaustive search
  - scalable :
    - logarithmic local state
    - logarithmic number of hops
  - fault-tolerance
    - periodic scanning
    - replication
  - drawbacks
    - only exact match queries
    - no locality awareness

# P2P technologies

- Unstructured P2P approaches
  - flooding based
  - non-exhaustive searches
- Distributed Hash Tables
  - routing based
  - exhaustive search
  - scalable :
    - logarithmic local state
    - logarithmic number of hops
  - fault-tolerance
    - periodic scanning
    - replication
  - drawbacks
    - only exact match queries
    - no locality awareness

# Complex queries over DHT networks

- XML-based description of resources
- Database operations over DHTs
- Multi-dimensional queries
- Range queries
- Trie-based lookup
  - range queries
  - parallel processing, latency limited by the depth of the trie
  - Approaches
    - Skip Graphs (complexities)
    - Nodewiz (no fault-tolerance)
    - Prefix Hash Tree (static trie)
    - P-Grid (static trie)
    - locality awareness issue

# Complex queries over DHT networks

- XML-based description of resources
- Database operations over DHTs
- Multi-dimensional queries
- Range queries
- Trie-based lookup
  - range queries
  - parallel processing, latency limited by the depth of the trie
  - Approaches
    - Skip Graphs (complexities)
    - Nodewiz (no fault-tolerance)
    - Prefix Hash Tree (static trie)
    - P-Grid (static trie)
    - locality awareness issue

# Comparison with other approaches

| **Functionality** | **Skip Graphs** | **PHT** | **P-Grid** | **DLPT** |
|---|---|---|---|---|
| Insertion | $O(\log(n))$ | $O(D)$ | $O(log(\Pi))$ | $O(T_{max})$ |
| Lookup | $O(\log(n))$ | $O(\log(D)\log(N))$ | $O(log(\Pi))$ | $O(T_{max})$ |
| Range messages | $O(m\log(n))$ | $O(o)$ | $O(\Pi_R)$ | $O(m)$ |
| Range time | $O(\log(n))$ | $O(D)$ | $O(log(\Pi))$ | $O(T_{max})$ |
| Fault-tolerance | repair | DHT-based | replication | replication |
| Locality | - | - | - | greedy |