Application of TRIE data structure and corresponding associative algorithms for process optimization in GRID environment

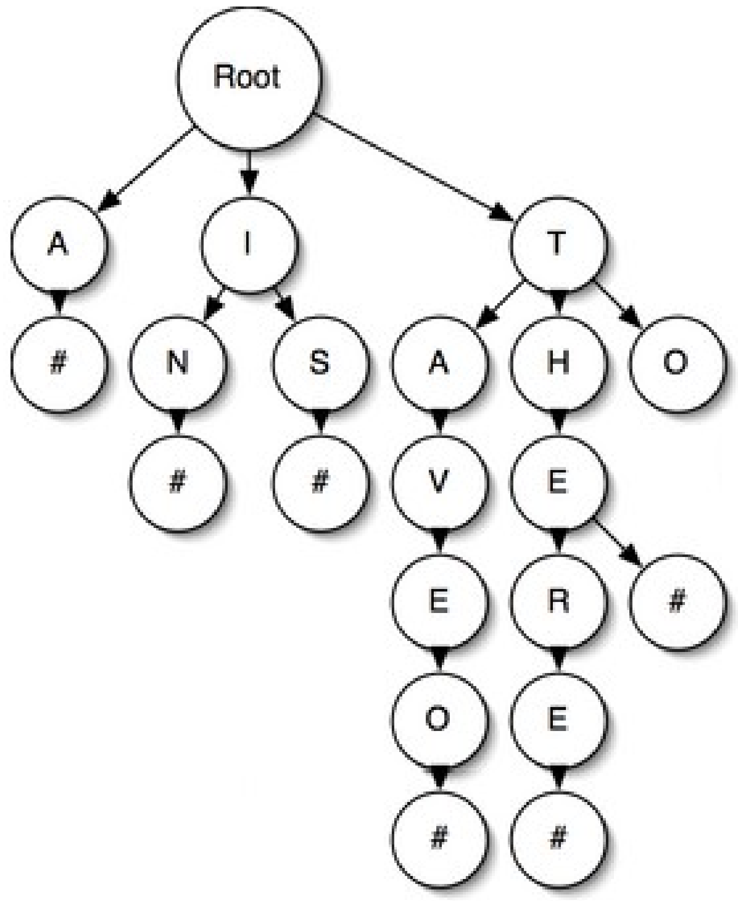
Kaftannikov I.L., Kashansky V.V., SUSU, Electronics Department

Dubna, JINR LIT

GRID'16

Introduction

- The main goal is to show **how TRIE** mechanisms **can influence operation of GRID environment**, delivery process of the resources and corresponding services, reducing the level of latency in various GRID environment sub-systems **at different levels of abstraction**.

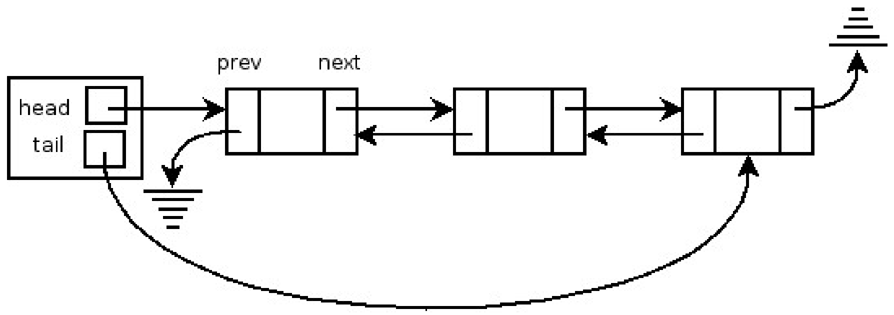


What is TRIE

- In computer science a **<u>TRIE</u>**, or a <u>prefix tree</u> is an ordered multi-way tree type data structure that is used to store byte arrays (strings, structures, dwords, qwords), and especially dictionaries, in extremely effective way.

- By Applying the Big O notation for lookup, insertion and deletion it becomes O(k) which can be written as **O(1)**. Thus these operations are performed in constant time irrespective of the n items, stored in TRIE.

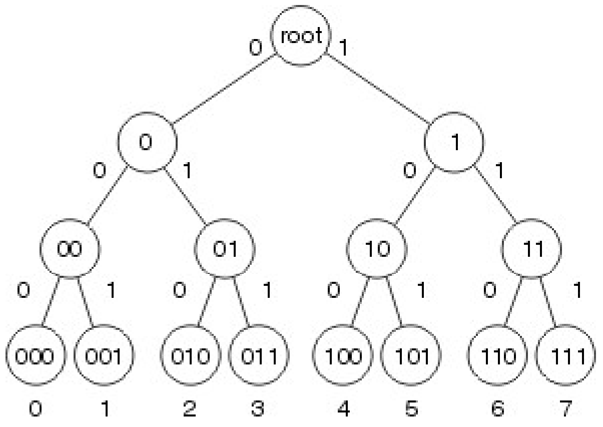
Straight solution. OSI L3

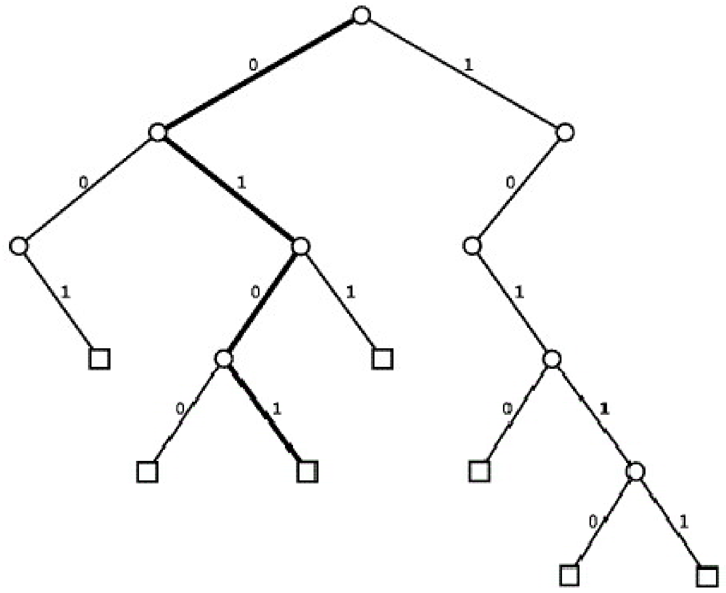


- The simplest way to search best suitable route for a given IP address is to view (search) records in a table stored in the memory as a linear data structure, for example, a linked list or array.

- The complexity of the algorithm in this approach will be **O(n)**, where n - number of entries in the routing table. A similar approach has been implemented in all hardware and “software” routers by default. Of course, this approach is not optimized for search time, but it does have its own advantages.

Bitwise TRIE solution. CFS technology





Diagrams above show a Bitwise TRIE. This tree is considered to store a sequence of **N bits as a key**. There is no key stored in the intermediate nodes, but an array of pointers is defined. Access to the subsequent elements of specific node is possible to get by index in this array respectively. 0 or 1. The access time is O (n), where n - the key length (in the worst case of 32 bits for IPv4).

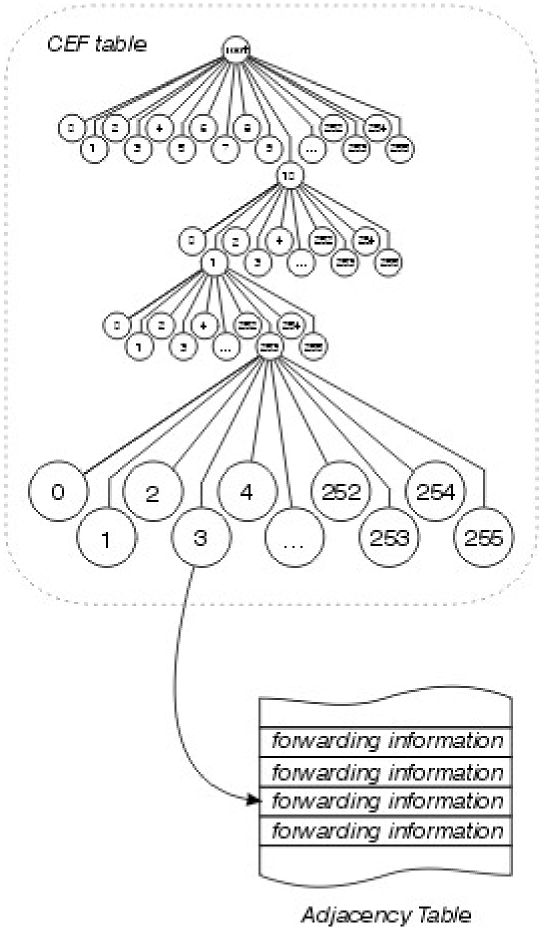
CFS technology key features

- This structure allows very fast retrieval of information from the cache. That in turn leads to lower latency;

But...

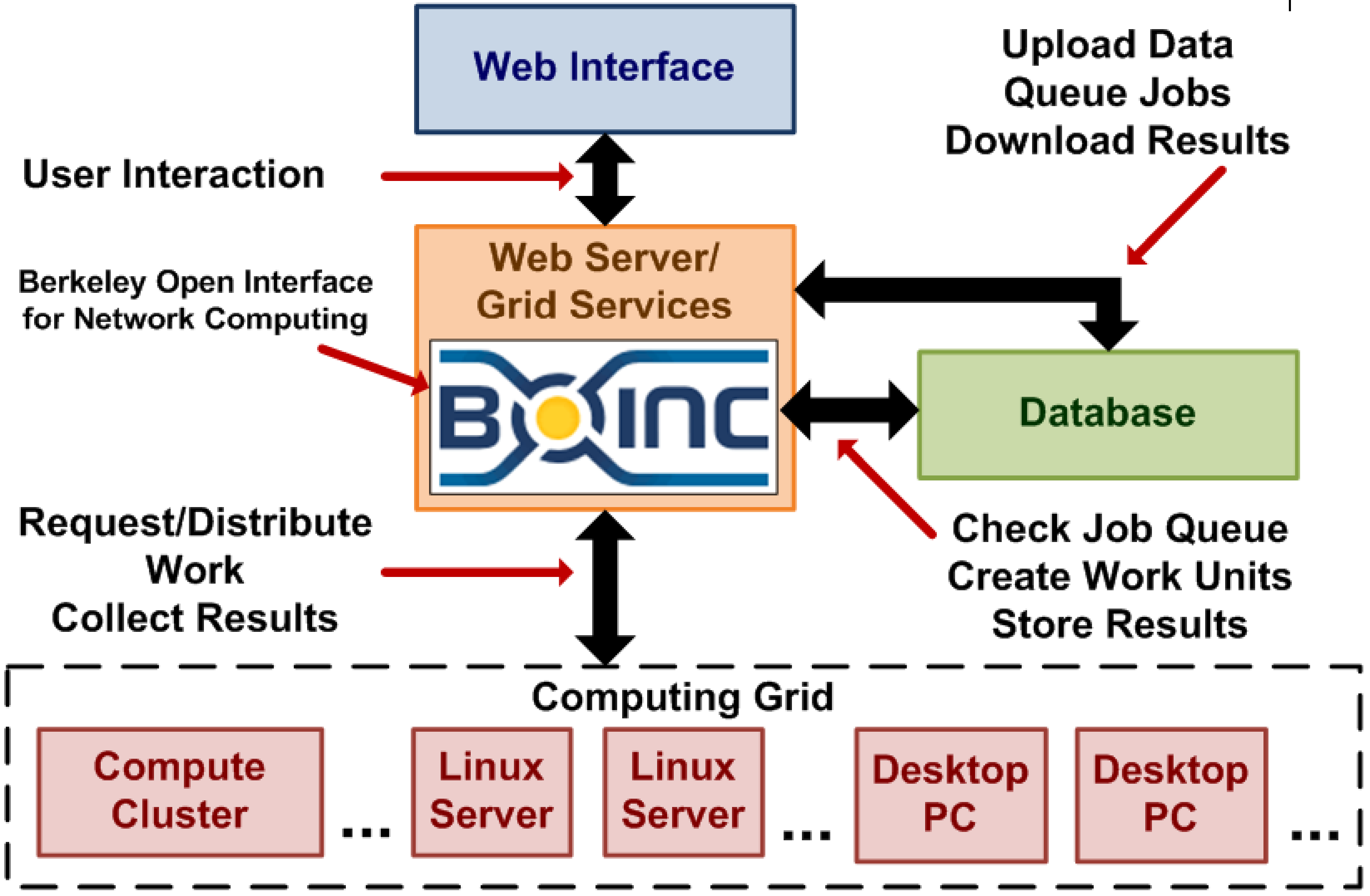
- The organization of storage requires additional CPU and RAM resources of router;
- When you change the information in the ARP tables, corresponding cache entries should be invalidated and removed;

Cisco Express Forwarding technology (CEF)

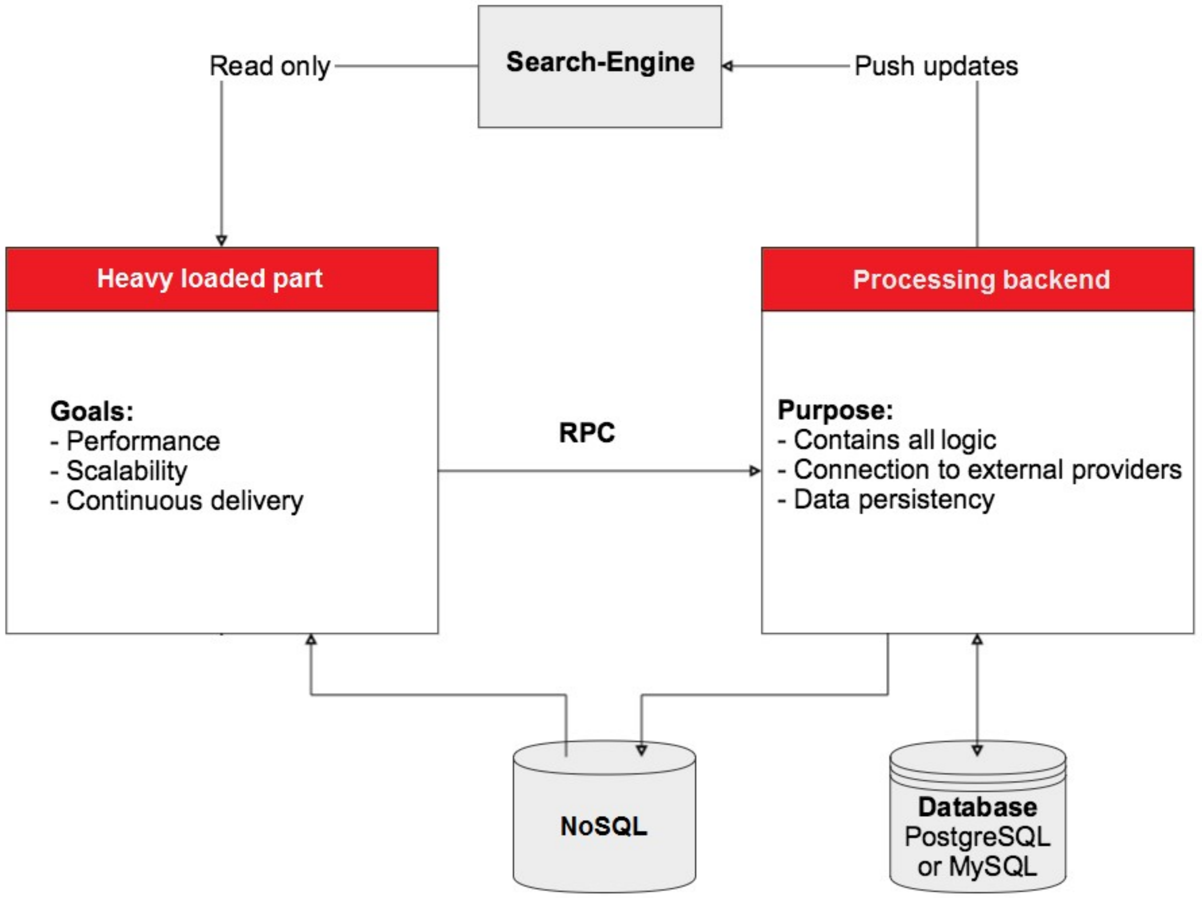


- Cisco Express Forwarding technology involves using of the **256 way TRIE** data structure known as well as the FIB (Forwarding Information Base), storing route (L3) information and a pointer to Adjacency Table. 256 way TRIE includes 4 levels of nesting, 256 (2^8, size of one IPv4 octet) options at each level. Endpoints (4 level) contain only pointers and the data stored in a separate structure.

BOINC and it’s architecture



TRIE as OSI L7 BOINC booster



- Avoid expensive SQL queries to a relational database with several joins and conditions;
- Performance boost is provided by a simplified software design without the architectural overhead of layers.

Experimental data output

N	2822	8512	13223	19357
T*10^{-3} c	1014	2449	3742	6208

N – Items count, T milliseconds spent to access

N (key)	8	16	24	32
n*(1/f)	46	86	119	147

N – Key's bits count, n – CPU clock spent, f –CPU clock frequency

Linear storage summary

Using “brute force” over linear storage gives:

- Less RAM resource consumption (data size considered to be the same);
- Search time is proportional to the size of the table;
- Significant consumption of CPU resources, which is proportional to entry count.

TRIE storage summary

Using “extraction by key” over TRIE storage gives:

- Reduction of the access time;
- Avoid cross-node comparisons of keys and storing them in nodes;
- Avoid tree balancing problem;
- Avoid hash table collisions;
- More efficient spending RAM and CPU resources (compared to other ADT). The CEF technology compared with CFS reduces the load on the CPU and RAM because of a more optimal allocation of nodes and keys modified structure.

Thank you!