Προχωρημένη Κατανεμημένη Υπολογιστική

HY623

Διδάσκων –
Δημήτριος Κατσαρός

@ Τμ. ΗΜΜΥ
Πανεπιστήμιο Θεσσαλίας

Διάλεξη 7η

A-tree

Distributed indexing of multidimensional data for cloud computing environments

Introduction

- Need for fast and efficient processing (even for clouds) of huge volume datasets
- Need for index structures
- MUST properties of index structures for clouds
  - Distributed
  - Space efficient
  - Support point and range queries
- Data belong to high dimensional space

Related work

- DHT & RT-CAN
- Distributed B-tree
  - Supports only point queries
- BR-tree
  - Nodes as p2p network
- EEMINC (Extended Efficient Multi-dimensional Index with Node Cube)
  - State-of-the-art for our problem
  - Master nodes each with R-tree (global index)
  - Slave nodes each with KD-tree

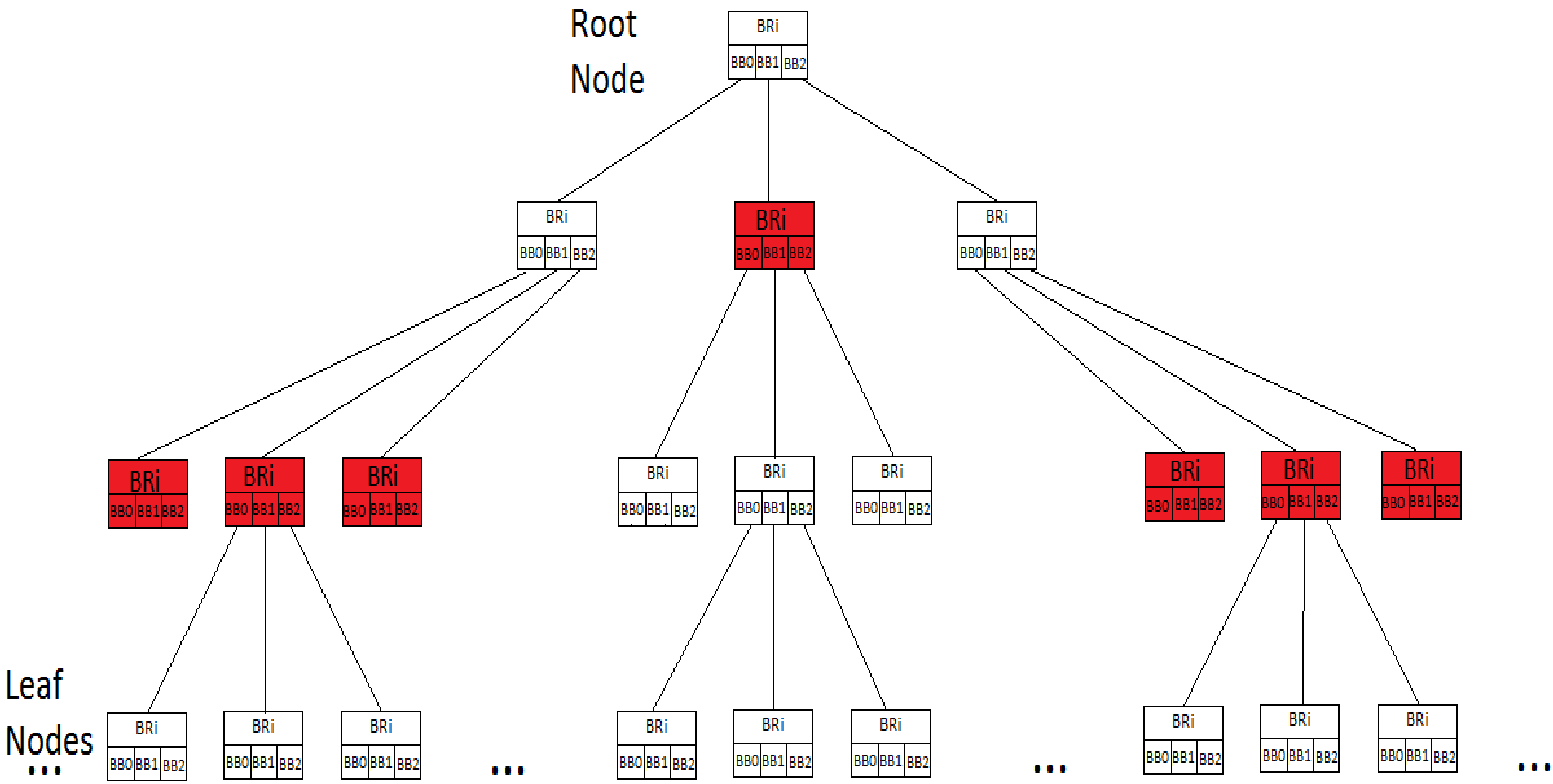
A-tree's architecture

- Nodes: Masters & Slaves
- Users
  - Query master nodes
  - Master nodes forward queries to appropriate slave nodes
- Slave Nodes return the result set to the users
- Query processing
  - Master node
    - Locates the relevant slave nodes
    - Forwards the query to these slaves
  - Local processing at slaves to build the result set
- A table is used for the global index @ master nodes
- For local index at slave nodes:
  - Bloom Filter
  - R-Tree

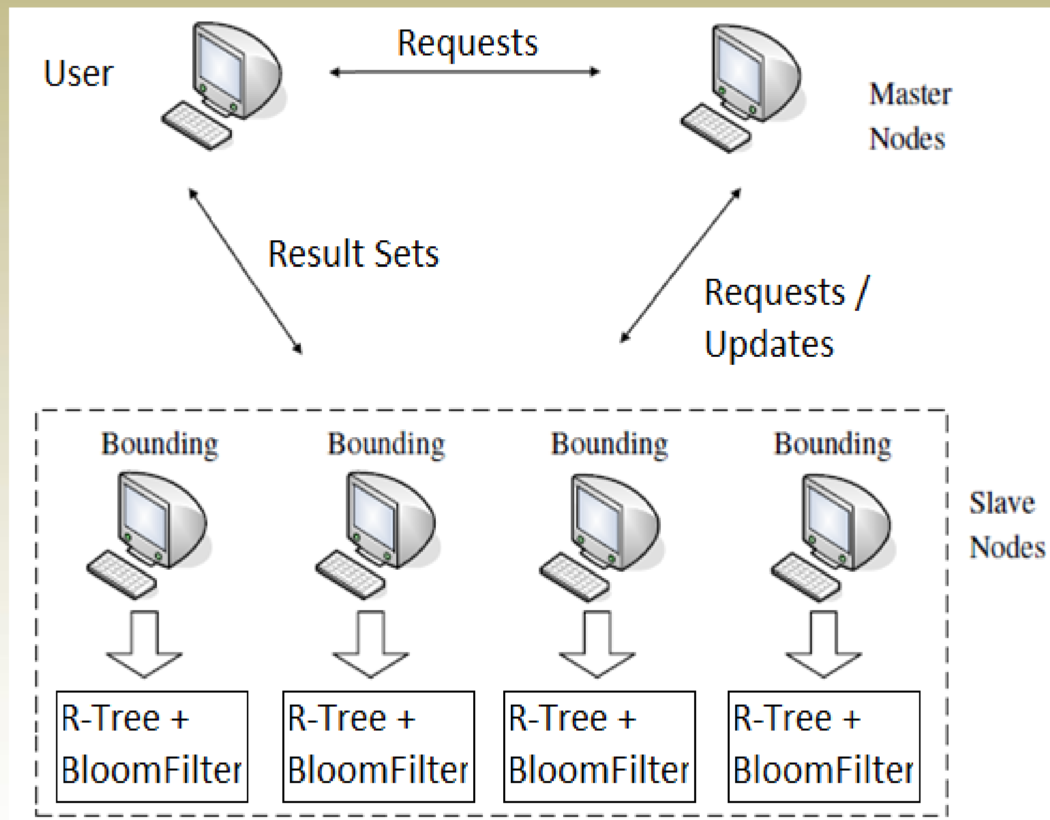
Indexing tree nodes

- An update is a combination of some hyper bounding boxes from R-tree's nodes
- We have a tradeoff between how many nodes to index, as more nodes will result to higher resource consumption on master nodes for finding relevant nodes to a query
- An update should be kept to minimal in order to:
  - Lower search cost at master nodes
  - Lower bandwidth usage
- Bloom filter is included in the update

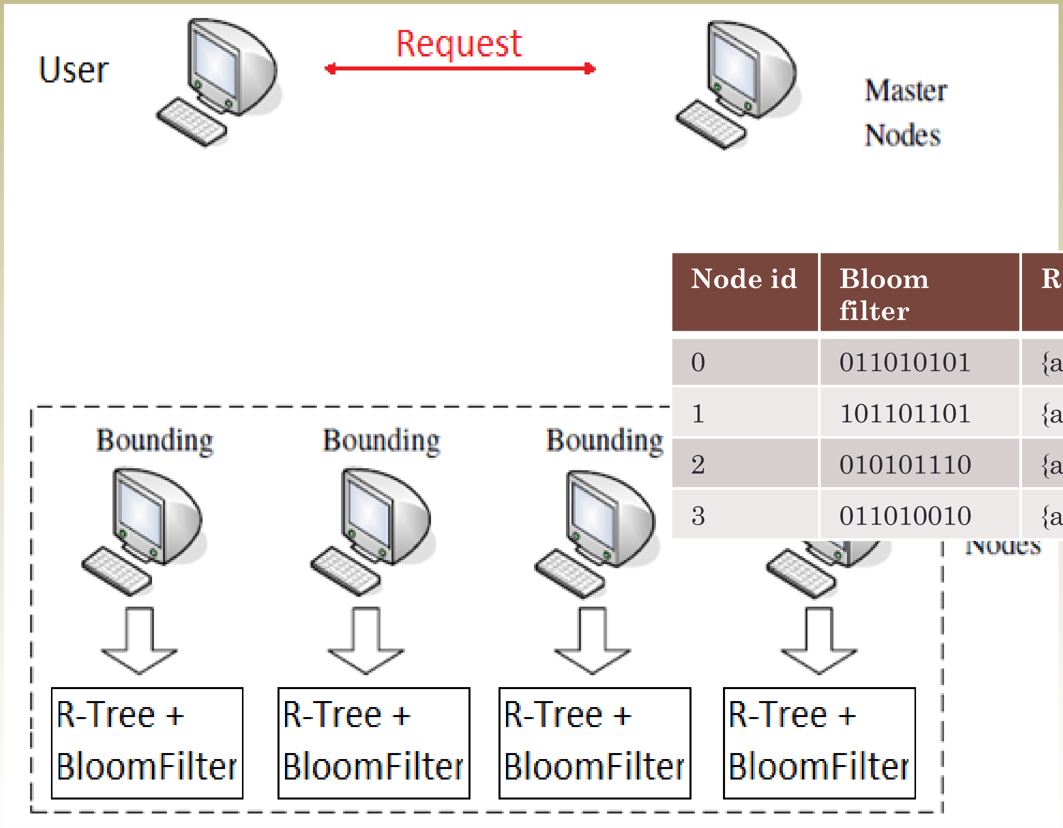
Indexing tree nodes



Processing queries

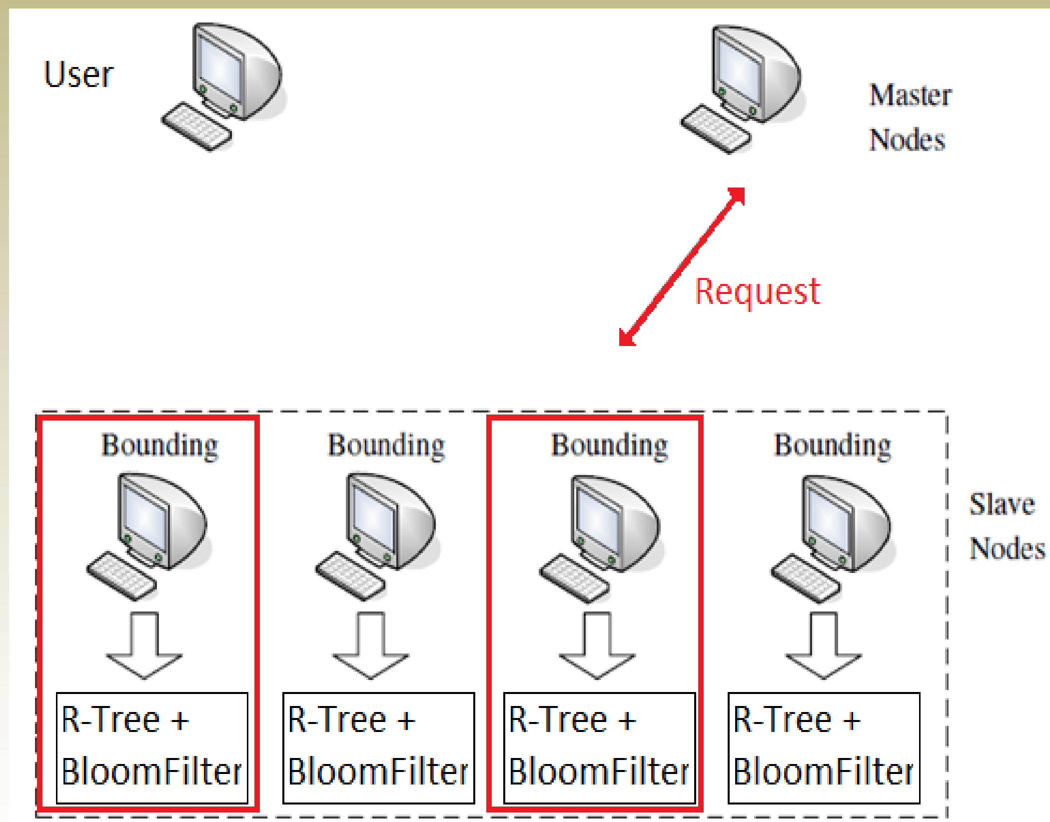


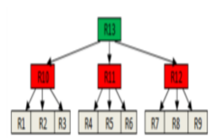
Processing queries



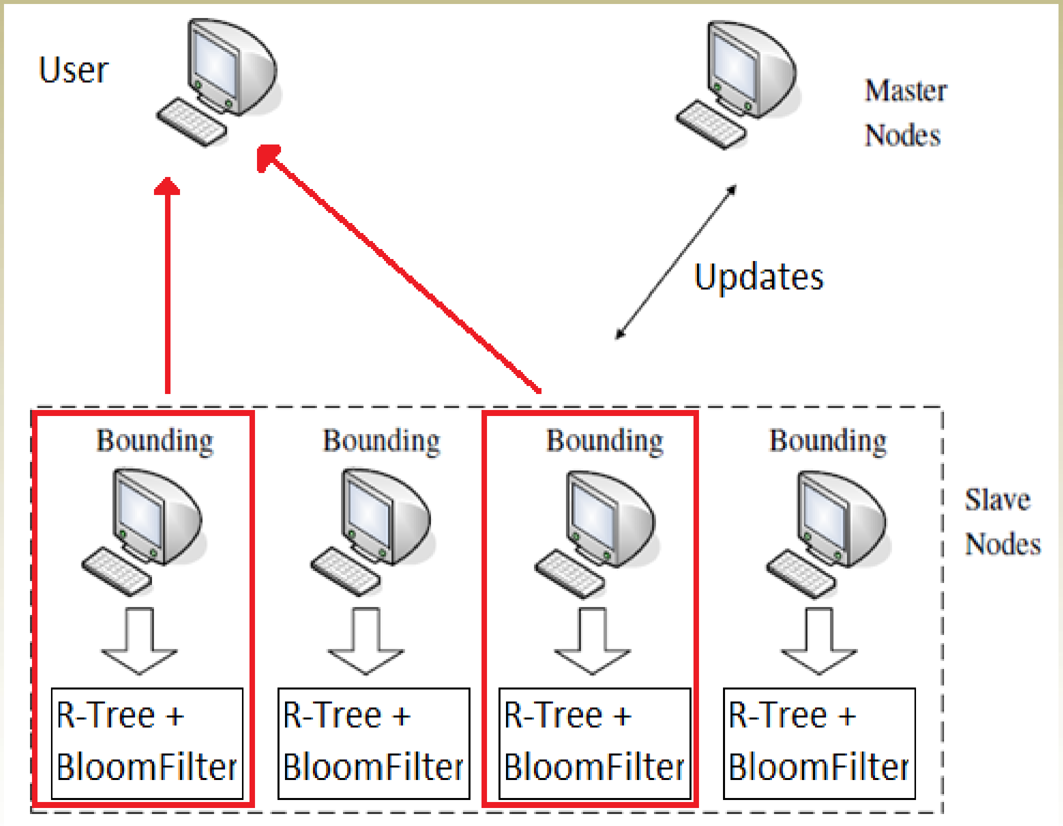
Node id	Bloom filter	Range
0	011010101	{a,b,c}-{x,y,z}
1	101101101	{a,b,c}-{x,y,z}
2	010101110	{a,b,c}-{x,y,z}
3	011010010	{a,b,c}-{x,y,z}

Processing queries

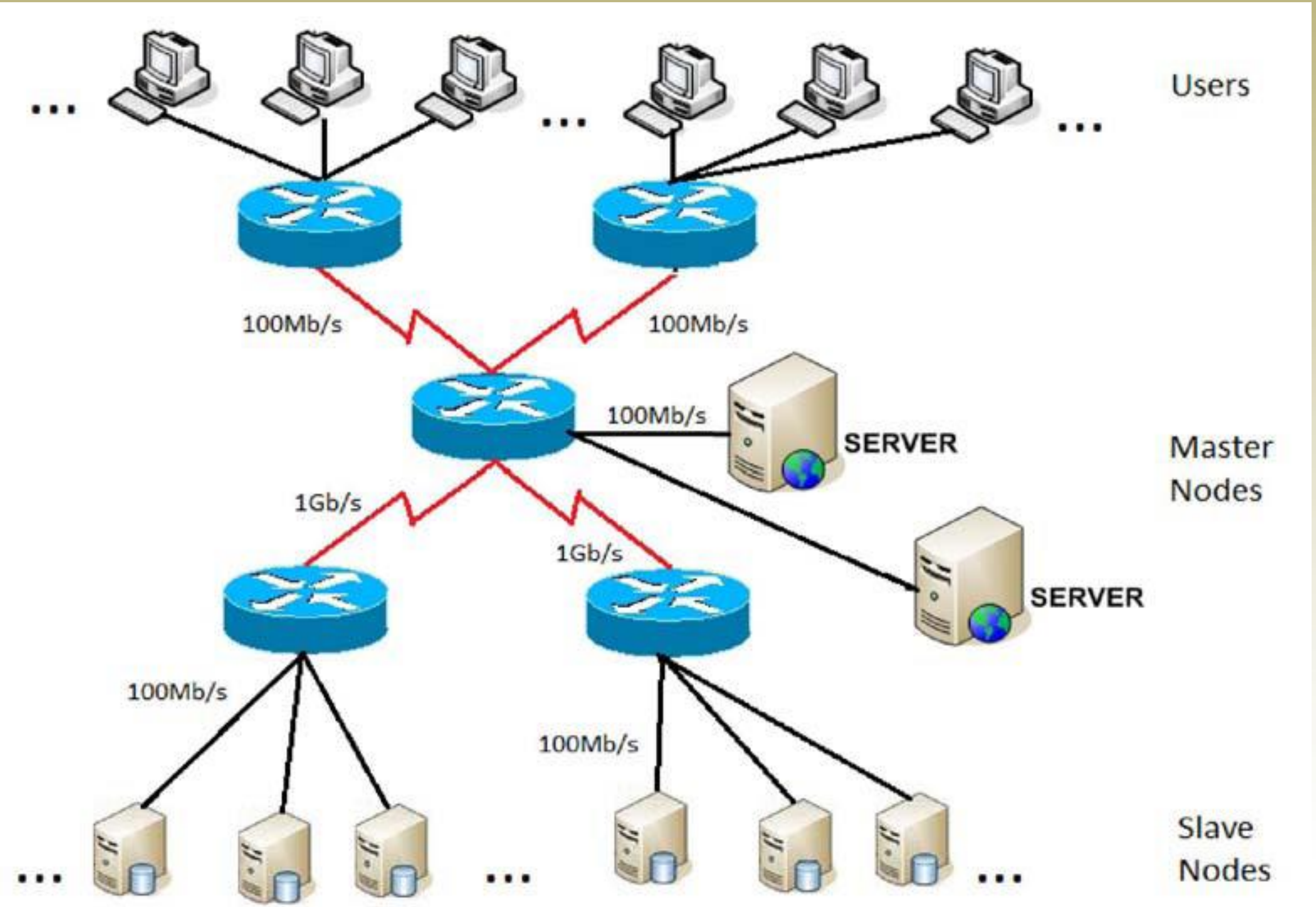




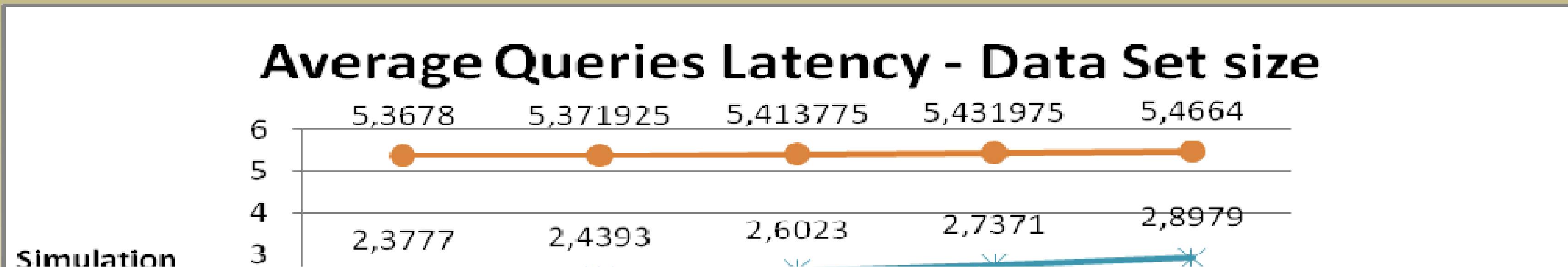
Processing queries



Experimental evaluation network topology

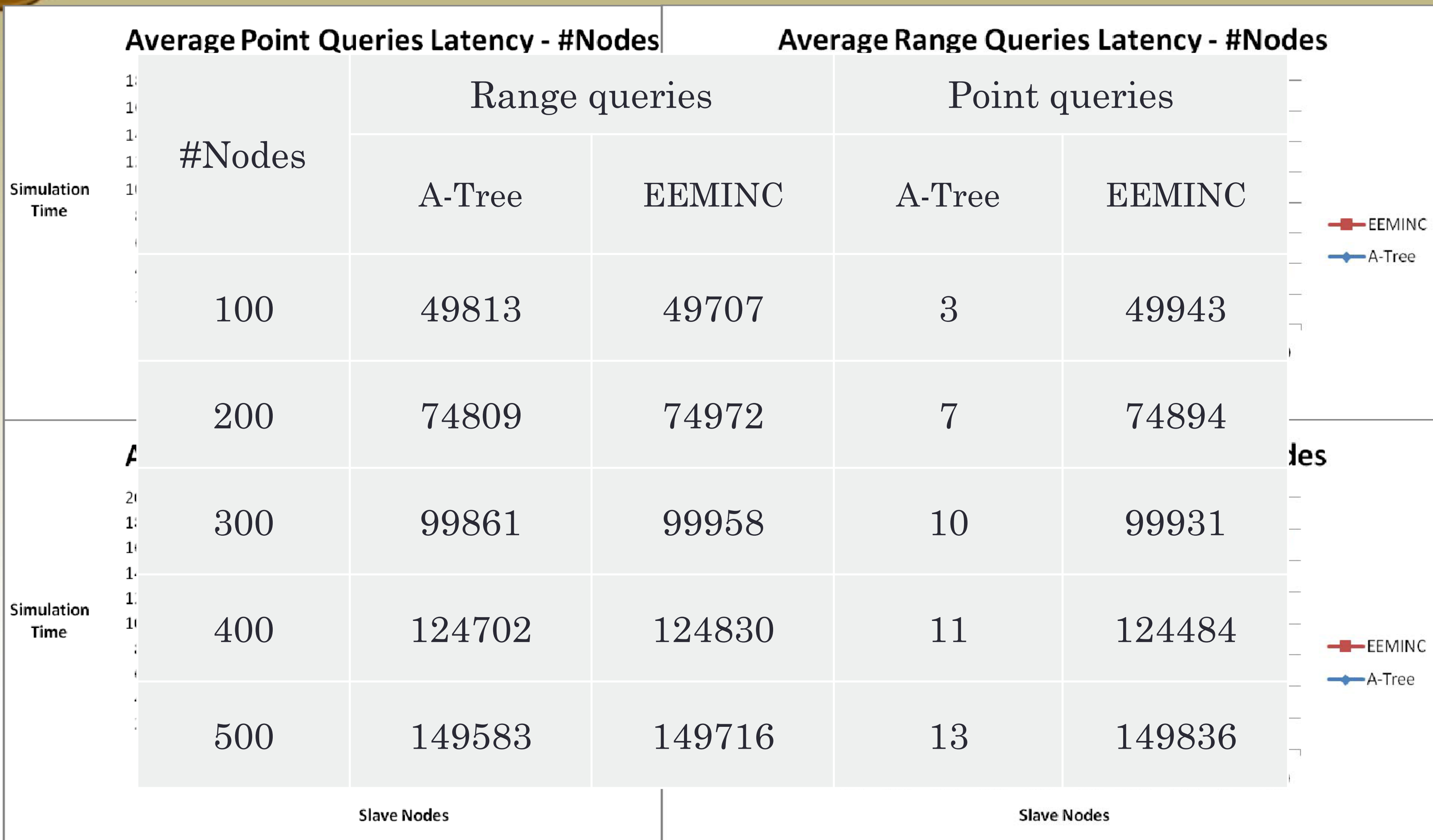


Experimental evaluation



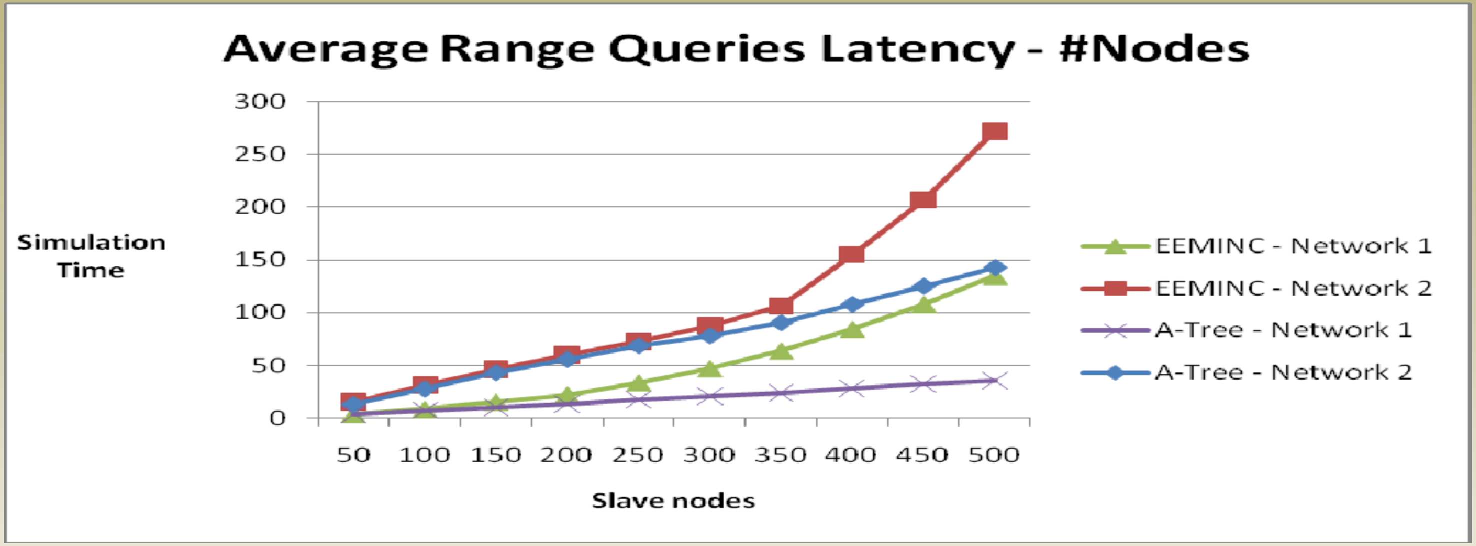
#Records/Node	Range queries		Point queries	
	A-tree	EEMINC	A-tree	EEMINC
15000	37325	37339	199	37434
30000	37189	37212	707	37500
45000	37049	37123	1509	37500
60000	36911	36918	2516	37481
75000	36791	36803	3577	37499

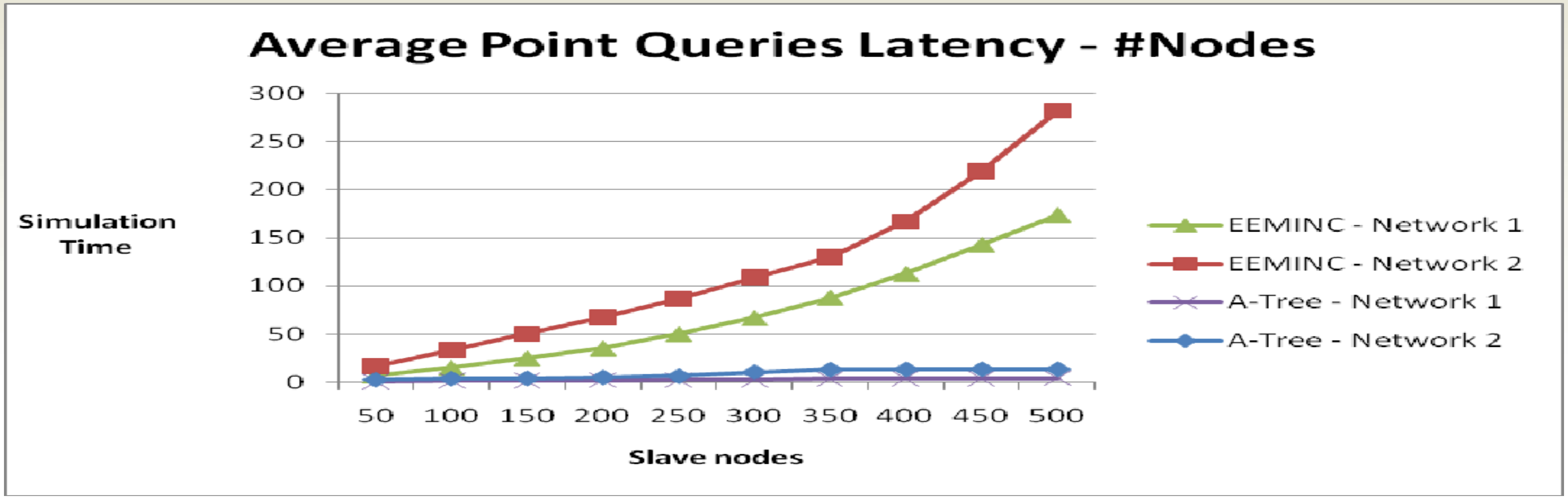
Experimental evaluation



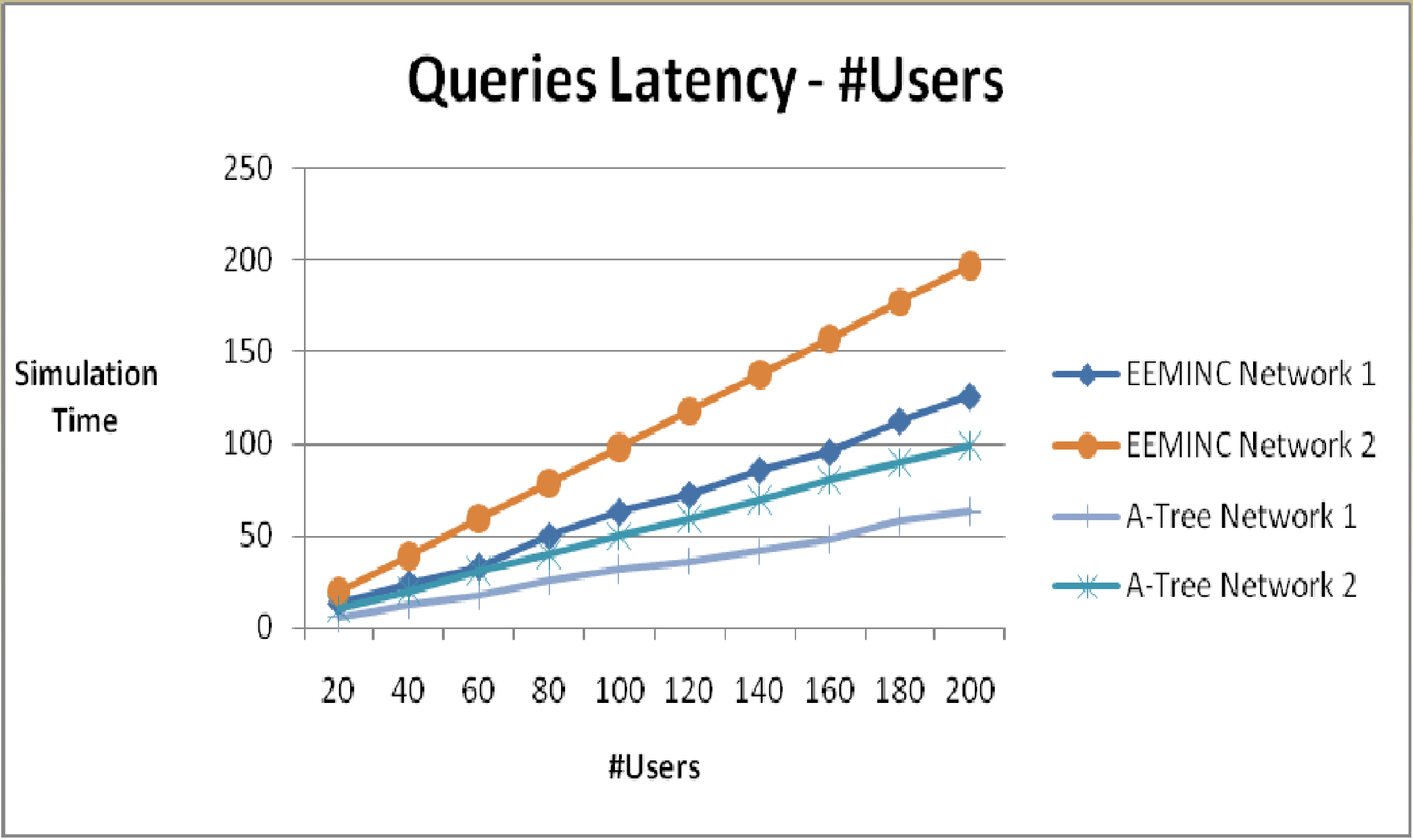
#Nodes	Range queries		Point queries	
	A-Tree	EEMINC	A-Tree	EEMINC
100	49813	49707	3	49943
200	74809	74972	7	74894
300	99861	99958	10	99931
400	124702	124830	11	124484
500	149583	149716	13	149836

Experimental evaluation





Experimental evaluation



Conclusions

- Need for cloud data indexing
- A-tree
  - R-tree & Bloom filters @ slaves
  - Array @ masters
- A-tree is particularly good for point queries
- A-tree scales linearly with system's growth
- Network usage is minimized
  - Performs very well for slow network configurations

Cloud migration decisions I

To lease CPUs from cloud OR buy CPUs?

Lease-or-buy decisions

- Johnson & Lewellen ("Analysis of Lease-or-Buy Decision," J. Finance, vol. 27, no. 4, 1972, pp. 815-823) suggested modeling the lease-or-buy decision as a capital budgeting problem

$$NPV = \sum_{T=1}^{Y} \frac{(P_T - L_T) - t(P_T - L_T - D_T)}{(1+k)^T} + \frac{S - t_g(S-B)}{(1+k)^T} - A,$$

- P_T = cash revenue expected from the use of the asset at year T
- L_T = pretax cash cost required to operate the asset at year T
- D_T = depreciation charge for year T
- k = cost of capital
- A = cash purchase price of the asset
- S = expected salvage value of the asset at the end of its life
- B = expected book value of the asset at the end of useful life
- t = corporate income tax
- t_g = tax rate for gain or loss on disposal of the asset

- The first term approximates the net after-tax operating profit
- The second term approximates the after-tax proceeds from asset salvage after its retirement.
- The third term incurs the asset's initial purchase cost

The time value of CPU

- This principle basically states that an investor always prefers to receive some fixed amount of money today rather than in the future
- Hence, a popular technique for making lease-or-buy decisions involves comparing investment cash flows at their present value by discounting future cash consumption with a rate of interest

profit-cost

$$PV = \frac{FV}{(1+k)^T} \quad \Rightarrow \quad NPV = \sum_{T=0}^{Y-1} \frac{C_T}{(1+k)^T}$$

Present and future capacity of a CPU

- Moore's law: integrated circuit transistors are expected to double approximately every two years
  - Its generalization to the microprocessor industry is that CPU performance is also expected to double every two years
- If Moore's law still describes the domineering CPU depreciation trend, the future capacity (FC) of a T-year-old CPU can be discounted to its present capacity (PC) through the biennial halving of CPU performance as follows:

$$PC = \frac{FC}{(\sqrt{2})^T}.$$

Net Present Capacity (NPC)

- Assuming the total useful capacity (TC) represents the expected CPU hours users consume annually in the cluster (for example, a 512-CPU cluster with 40 percent utilization provides a TC of $512 \times 365 \times 24 \times 0.4$ CPU hours per year), from previous Equation we can calculate a cluster's net present capacity (NPC) over an operational life span of Y years as follows:

$$NPC = TC \times \sum_{T=0}^{Y-1} \left(\frac{1}{\sqrt{2}}\right)^T \Rightarrow NPC = TC \times \frac{1-\left(\frac{1}{\sqrt{2}}\right)^Y}{1-\frac{1}{\sqrt{2}}}$$

The cost R of a CPU hour

- We can define as the cost of CPU hour the following quantity: $R = \frac{NPV}{NPC}$
- *Purchase case*: R(purchase) = $\frac{(1-\frac{1}{\sqrt{2}}) \times \sum_{T=0}^{Y-1} \frac{C_T}{(1+k)^T}}{(1-(\frac{1}{\sqrt{2}})^Y) \times TC}$
- *Lease case*: R(lease) = $\frac{\sum_{T=0}^{Y-1} \frac{C_T}{(1+k)^T}}{Y \times TC}$

No depreciation in the computational capacity because the lessee can always acquire the latest IT capacity from a competitive market over the operational life span of Y years. Thus: **NPC = Y × TC**

The cost R of a CPU hour

- Purchasing a cluster and upgrading it annually with the newest CPU to avoid the performance degradation cost
- Annualized operating cost includes repurchasing new CPUs (assum: price is equivalent to the server cluster's original purchase price)
- Thus: $$NPV = C_0 + \sum_{T=1}^{Y-1} \frac{C_T - A}{(1+k)^T}$$
- Because I'm annually upgrading the purchased server cluster, the same CPU performance degradation of an aging cluster isn't a factor (similar to the leasing case)
- *Purchase-Upgrade*: R(purchase-upgrade) = $$\frac{C_0 + \sum_{T=1}^{Y-1} \frac{C_T - A}{(1+k)^T}}{Y \times TC}$$

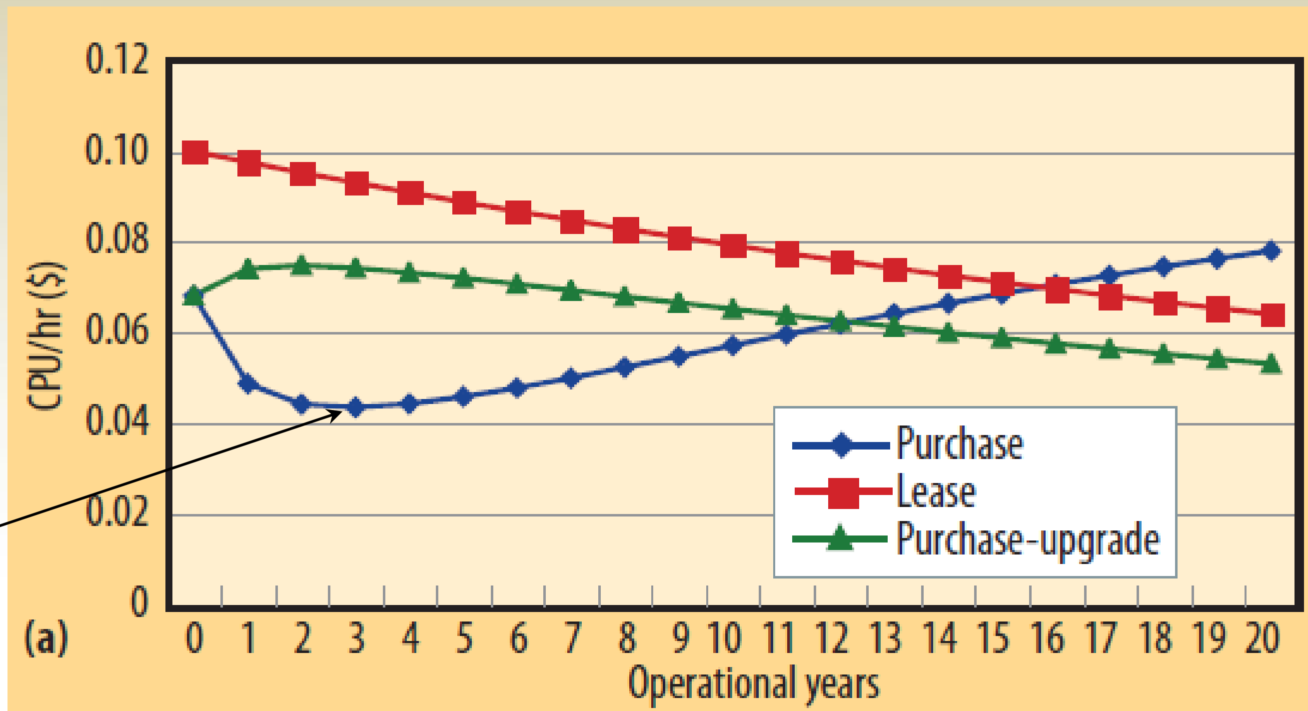
Example

System

- 5,000 quad-core processors: 60000 CPUs
- k= 5%
- Unavailability once per week
- 99 percent operational reliability
- 100% CPU utilization

TCPU = 60,000 CPUs
TC= TCPU × H × μ = 60,000 × ((365 − 52) × 24) × (0.99 × 1.0) = 440 million CPU hours annually

For leasing: $0.10 per CPU hour



OPT