Influence Maximization in Social Networks and Big Data Programming Principles

Hwanjo Yu
POSTECH
http://dm.postech.ac.kr

flickr

Over 1 Billion SNS users !!

twitter

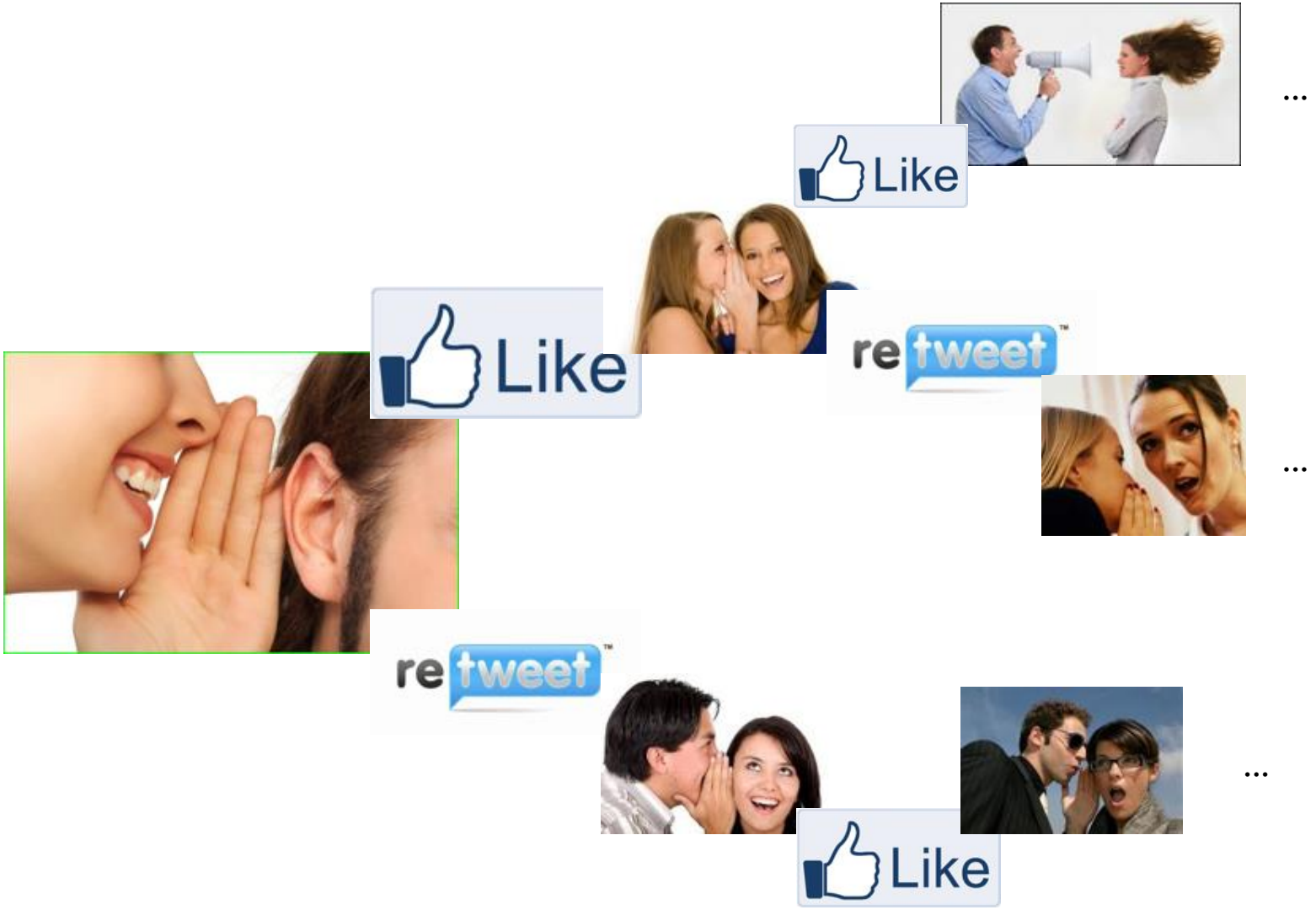
LinkedIn

facebook

December 2010

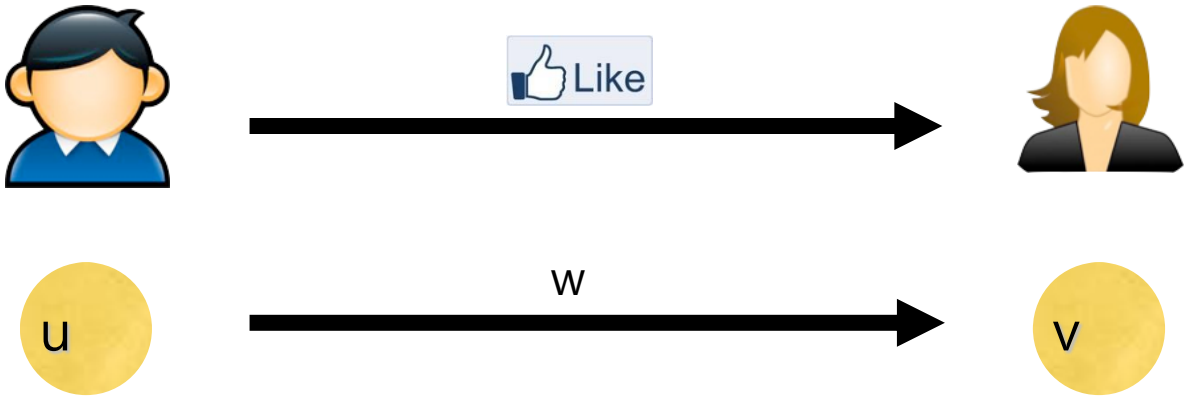
Viral Marketing

"Word-of-Mouth" Effect > TV advertising



Abstracting Social Networks

$G(V, E, W)$



Quantifying Influence

The expected number of entities influenced by a node set S

Influence Maximization

- GOAL : finding the ***k*** most influential individuals in social networks

$$\arg \max_{S \subseteq V, |S| = k} \sigma(S)$$

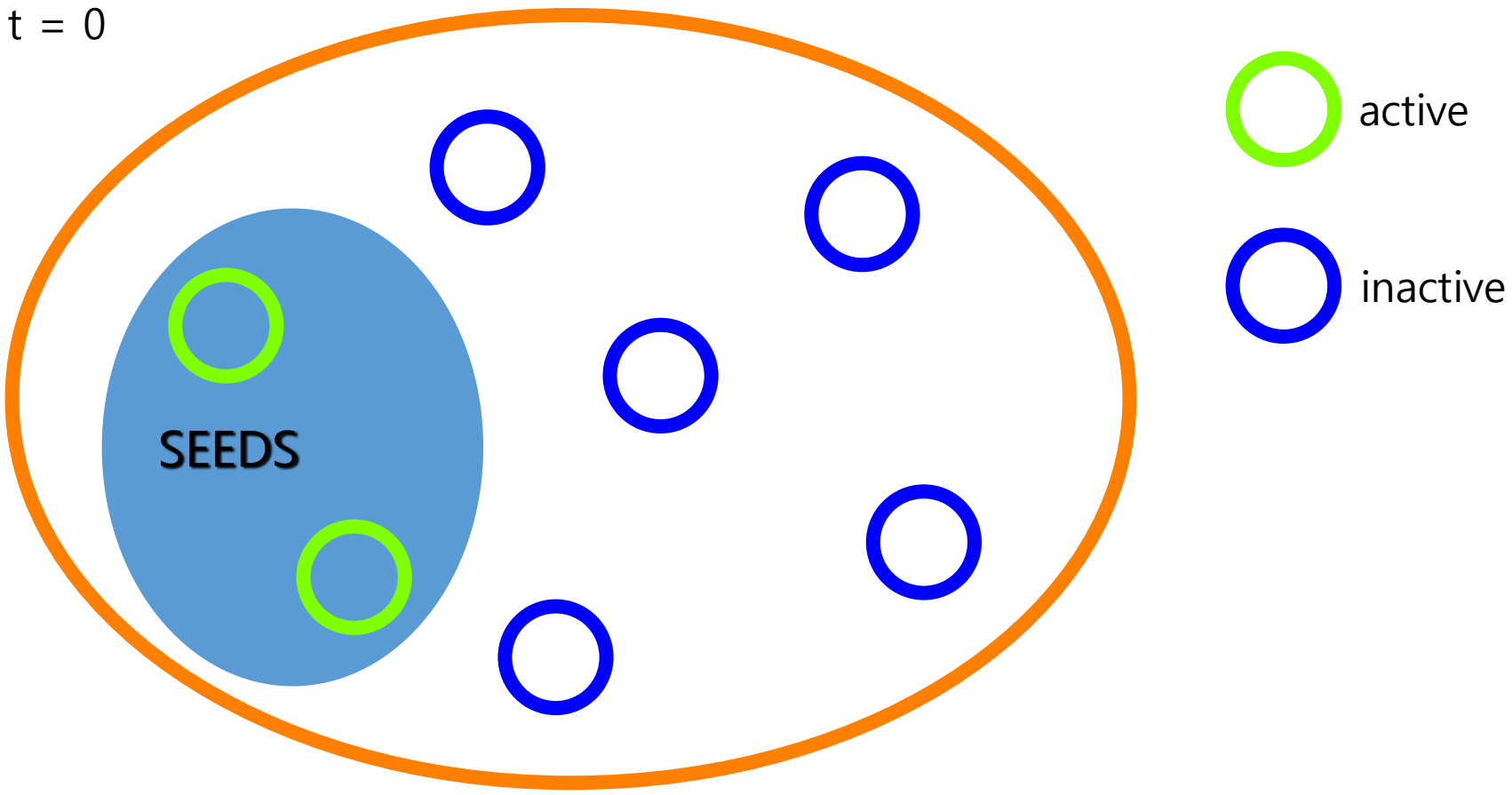
s on …

The expected number of entities influenced by *S*
DEPENDS ON
how influence is propagated through a graph

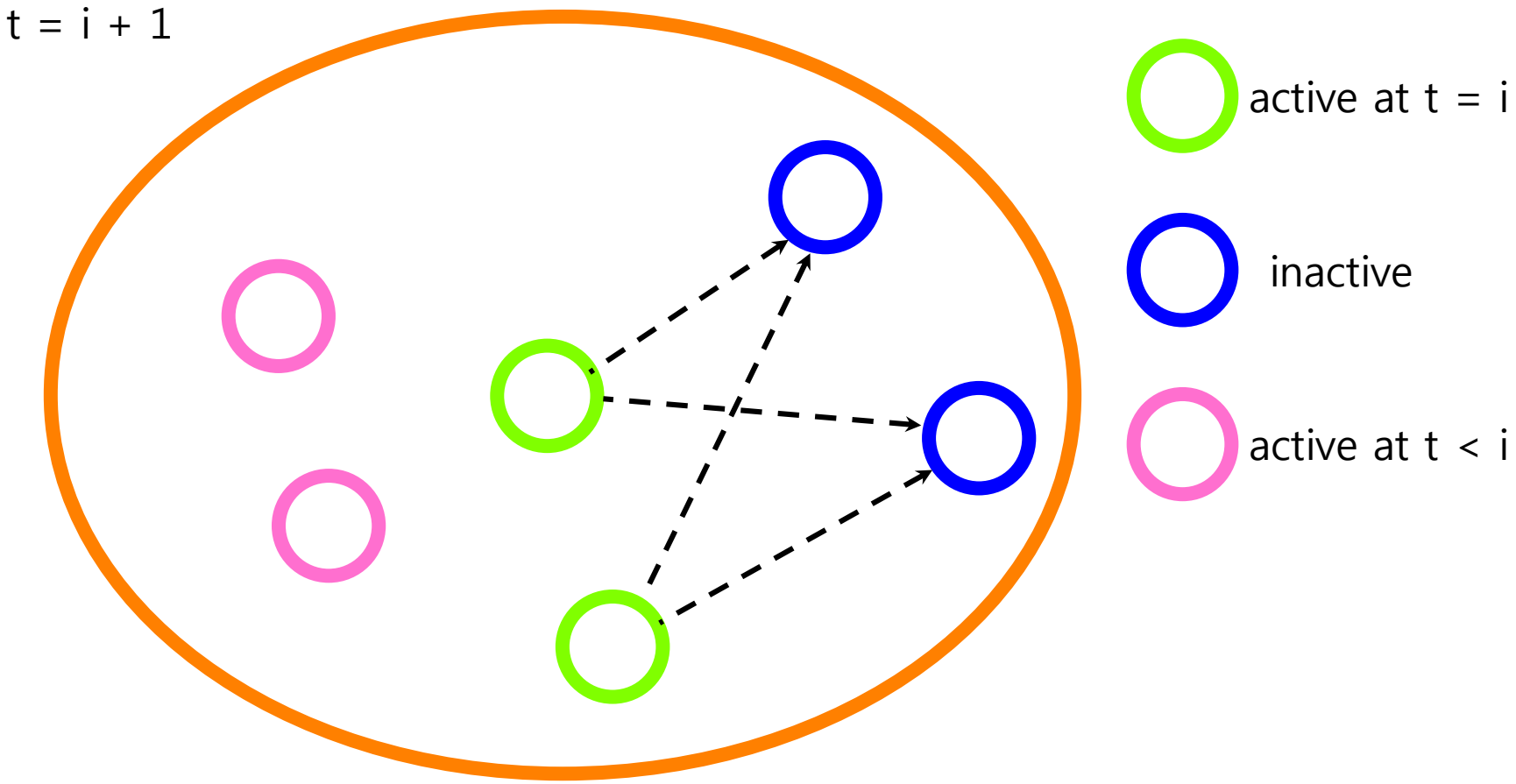
Influence Diffusion Model

IC and LT model (Kempe 2003 KDD)
IC-N model (Chen 2011 SDM)
CT-IC model (Lee 2012 ICDM)
…

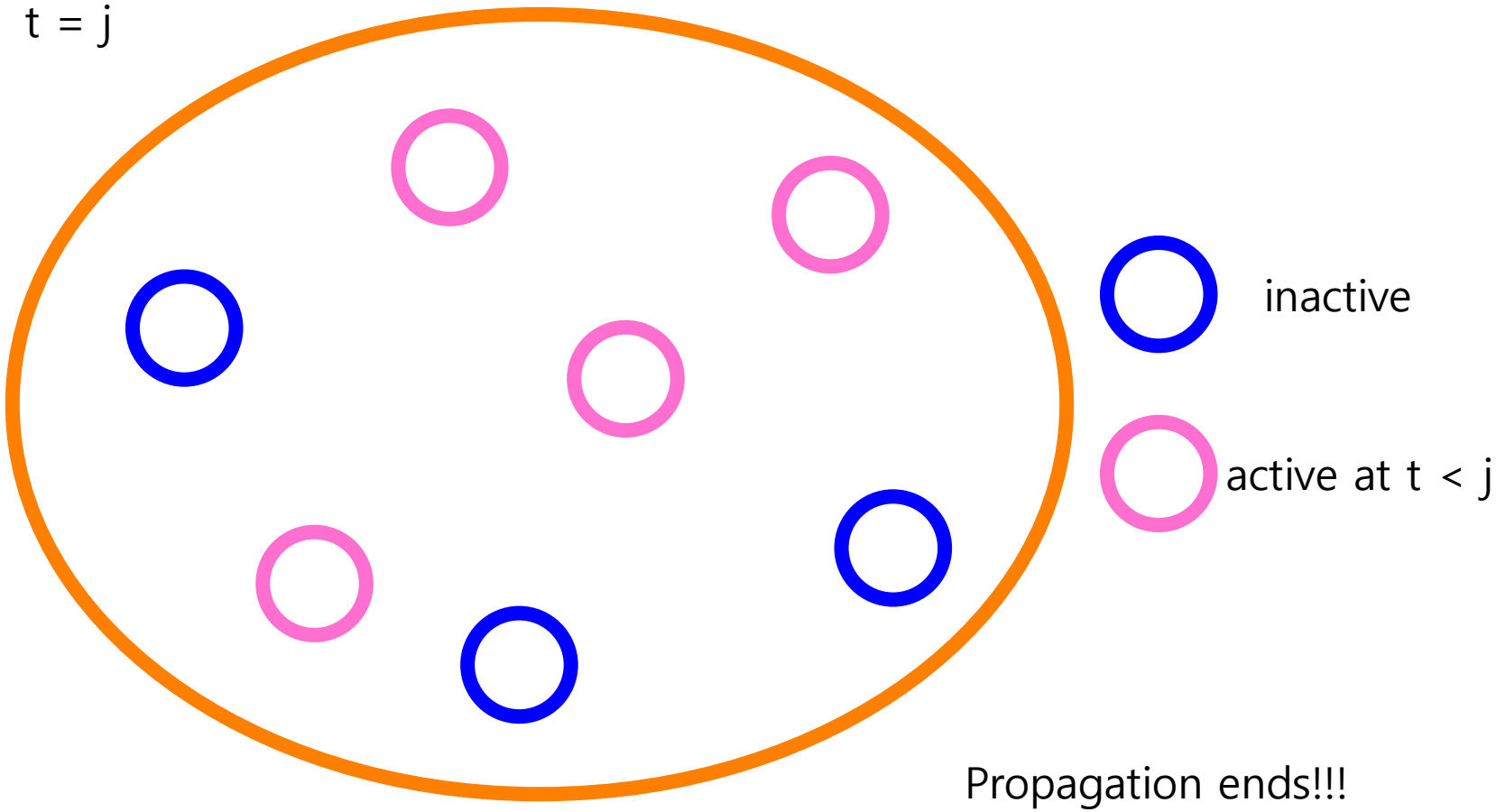
Independent Cascade (IC) model



Independent Cascade (IC) model



Independent Cascade (IC) model



Micro Level

$$\arg \max_{S \subseteq V, |S| = k} \sigma(S)$$

#P-HARD

Evaluating σ(S) with IC model

- **#P-hard**
- Heuristics
  - Monte-Carlo Simulation (KKT 03) → [1]
  - Shortest path between two nodes (KS 06) → [2]
  - Simultaneous simulation (CWY 09) → [1]
  - Breaking down a graph into communities (WCS 10) → [1]
  - **PMIA**: Local arborescence based on the most probable path (CWW 10) → [2] (**the state-of-the-art algorithm**)

  - [1] : too long processing time due to influence diffusion simulations
  - [2] : can be applied to only IC model

  - *IPA [ICDE 2013] : 10x times faster, accurate, extendible to other IC-based models, easily parallelized… (will be discussed later)*

Macro Level

$$\arg \max_{S \subseteq V, |S| = k} \sigma(S)$$

NP-HARD

Greedy Algorithm (KDD 03)

- Repeatedly select the node which gives the most marginal gain of σ(S)

Algorithm 1 Greedy(G, k)

1: $S = \phi$
2: **for** $i = 1$ **to** k **do**
3: $\quad u = \arg\max_{v \in V \setminus S} \sigma(S \cup \{v\}) - \sigma(S)$
4: $\quad S = S \cup \{u\}$
5: **end for**
6: **return** S

It guarantees approximation ratio **1 – 1/e**,
if σ(S) satisfies ***non-negativity, monotonicity, and submodularity***

IC (Independent Cascade) Model

Existing Models Ignore …

*An individual can affect others **multiple** times..*

Yesterday

Today

Tomorrow

Existing Models Ignore …

*Marketing usually has a **deadline**..*

Yesterday

Today

Tomorrow

CT-IC: Continuously Activated and Time-Restricted Independent Cascade Model for Viral Marketing [ICDM 2012]

1. Propose a new influence spread model, *CT-IC*, for viral marketing, which generalizes previous models such that
   - An individual can affect others multiple times.
   - Marketing can have a deadline.

2. Prove CT-IC model satisfies *non-negativity, monotonicity, and submodularity* and thus guarantees 1 – 1/e approximation ratio.

3. Harder to evaluate $\sigma(S)$ in CT-IC => PMIA does not work! => CT-IPA (an extension of IPA [ICDE 2013])

Three conditions for applying Greedy algorithm

- *Non-negativity* $\sigma(S) > 0$
- *Monotonicity* For $S \subseteq T, \sigma(S) \leq \sigma(T)$
- *Submodularity* For $S \subseteq T$, $\sigma(S \cup \{v\}) - \sigma(S) \geq \sigma(T \cup \{v\}) - \sigma(T)$
- **CT-IC** satisfies all three conditions

Theorem 1: The influence spread function $\sigma(\cdot, t)$ is monotone and submodular for all $t \geq 0$.

Dataset

Dataset	HEP	PHY	EPINION	AMAZON
Directedness	Undir	Undir	Dir	Dir
# of Nodes	15K	37K	76K	262K
# of Edges	59K	232K	509K	1235K
# of Connected Components	1781	3883	2	1
Average Size of Components	8.6	9.6	38K	262K
θ for CT-IPA	1/32	1/64	1/64	1/16

Model Comparison : IC vs. CT-IC

Table II
TOP-20 SEED NODES OF IC MODEL AND CT-IC MODEL SOLUTION.

(a) On PHY

IC model solution	4840	1568	5192	5120	7387
	12081	2356	10653	4115	23571
	3460	3808	969	809	5567
	2443	3566	5312	6342	3673
CT-IC model solution	4840	5192	5120	1568	809
	4115	2356	3460	23571	12081
	7132	**3842**	10653	**4109**	3673
	6342	**3712**	**2928**	**3982**	**2289**

(b) On AMAZON

IC model solution	17747	222839	25699	18076	168039
	18337	232448	7266	11129	45391
	176067	9657	64815	183084	27562
	59541	14461	238375	114241	1385
CT-IC model solution	17747	176067	**56415**	**51234**	**200657**
	238375	18076	**236670**	**259011**	222839
	6290	**205434**	**143531**	**199539**	**59541**
	25699	**178335**	**82533**	**114241**	**95315**

Model Comparison : IC vs. CT-IC

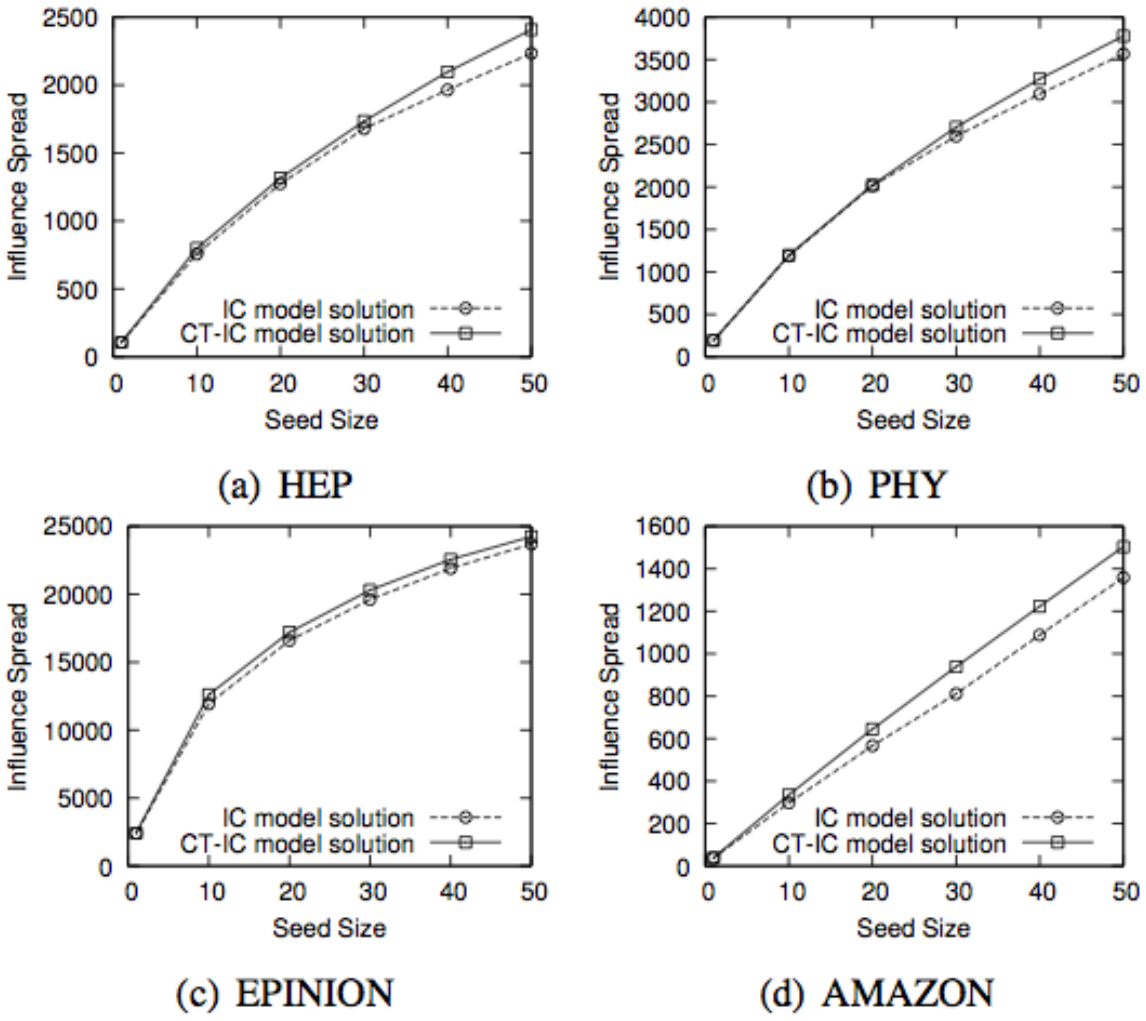


Figure 2. Comparison between IC and CT-IC models.

Effect of Marketing time Constraint

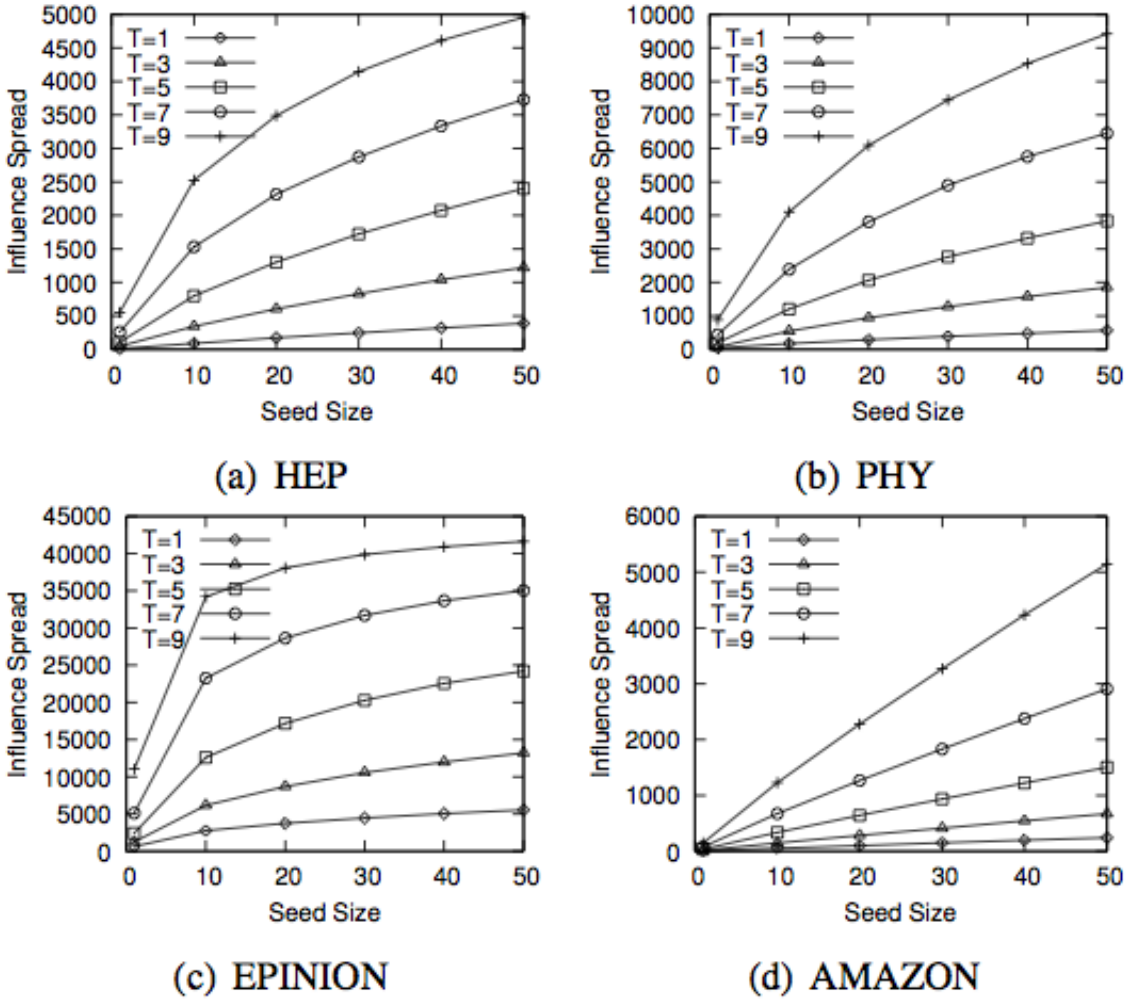


Figure 3. The change of influence spread with respect to T.

Influence Spread

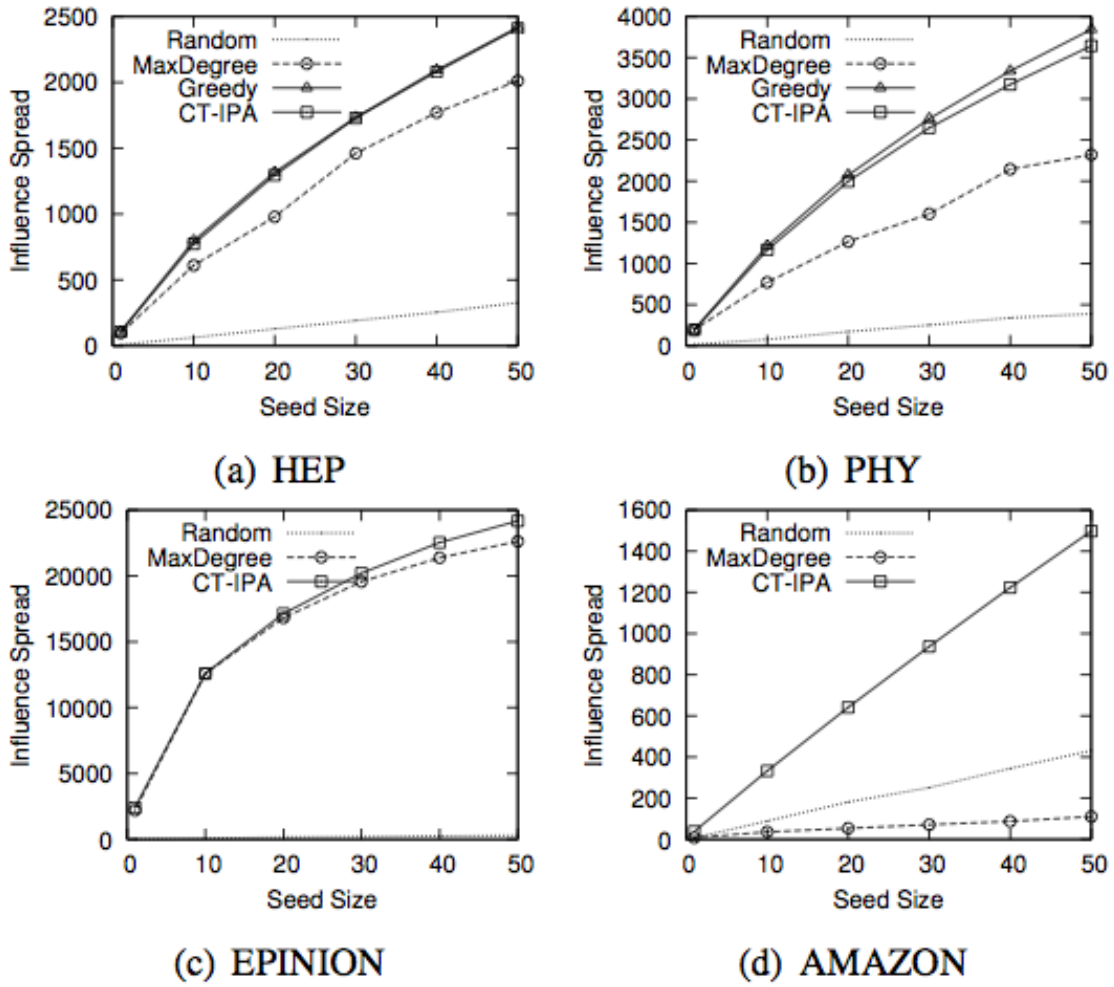


Figure 4. Influence spread of various algorithms.

Processing Time

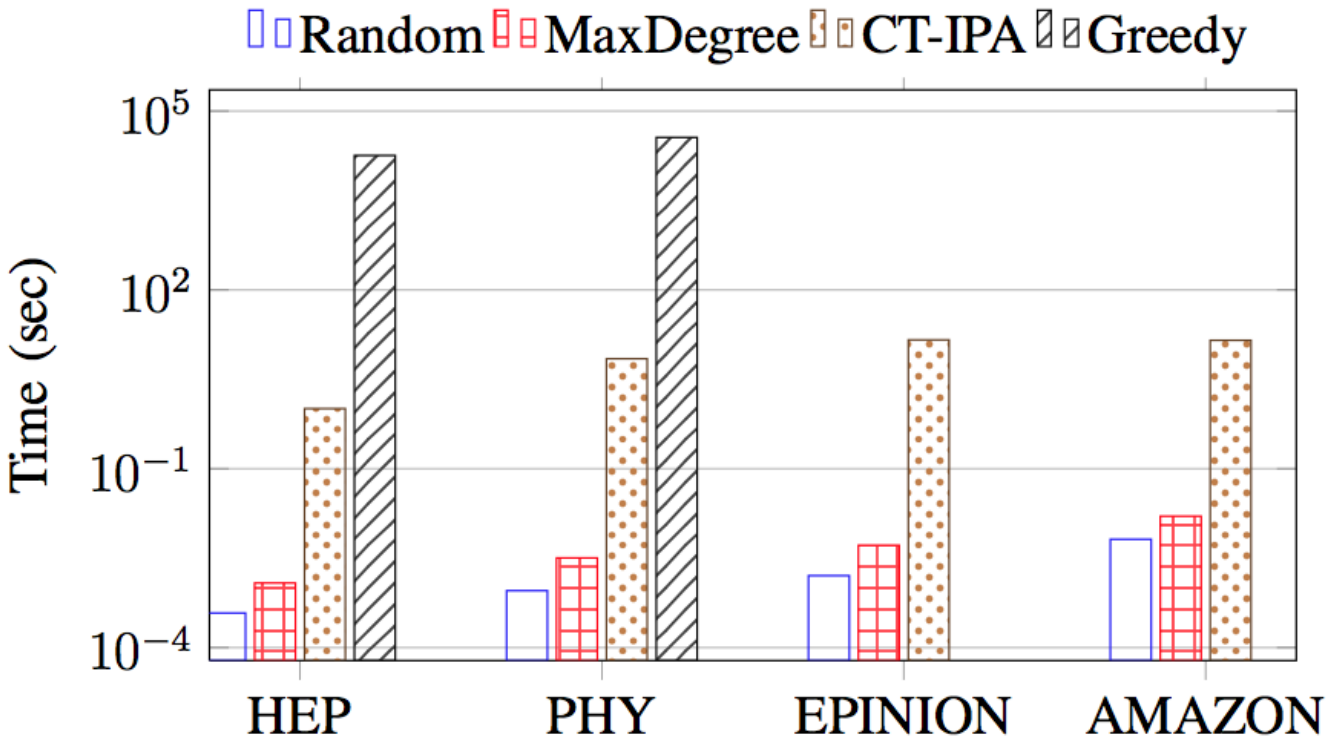


Figure 5. Processing time of various algorithms.

Finding most influential individuals in SNS =>

Influence maximization =>

NP-hard *in Macro &* ***#P-hard*** *in Micro*

The state-of-the-art algorithms for IC-based models =>

PMIA [KDD 2010] and **MIA-N** [SDM 2011].

IPA:

Scalable and Parallelizable Processing of Influence Maximization for Large-Scale Social Network [ICDE 2013]

IPA:
Scalable and Parallelizable Processing of Influence Maximization for Large-Scale Social Network [ICDE 2013]

1. **10x times faster** than PMIA (the state-of-the-art algorithm)

2. **Uses much less memory** than PMIA;
   - IPA successfully produces results on graphs of millions of nodes using 4GB memory where PMIA fails with 24GB memory.

3. **Accurately approximates** influence spread;
   - IPA's accuracy is close to that of Greedy solutions with 20k times MC simulation and is higher than that of PMIA overall.

4. **Can be applied to all IC-based models**;
   - PMIA cannot be applied to CT-IC model.

5. **Easily parallelized**;
   - The parallel IPA speeds up as # of CPU cores increases, and more speed-up is achieved for larger data sets.

IPA:
Scalable and Parallelizable Processing of Influence Maximization for Large-Scale Social Network [ICDE 2013]

- Key Ideas
  - Extremely localizing influence: evaluating influence based on path between two nodes
  - "Path" includes all meaningful(?) paths, not only the shortest paths => Use more memory initially but extremely simplify marginal influence computation => memory-efficient computation of CELF greedy

- Result
  - Fast
  - Accurate
  - Memory efficient
  - Flexible
  - Parallelizable

Intuition of IPA

- Extremely localizing influence
  - Influence path between two nodes as influence evaluation unit
  - Considering all path is not tractable (#P-hard)
  - Considering all meaningful path
    - Computing margin is simple
    - Save memory by holding the meaning paths of only top 3k nodes in the priority queue
  - PMIA considers only the shortest paths
    - Computing margin is complicated
    - Need to hold all pair shortest paths

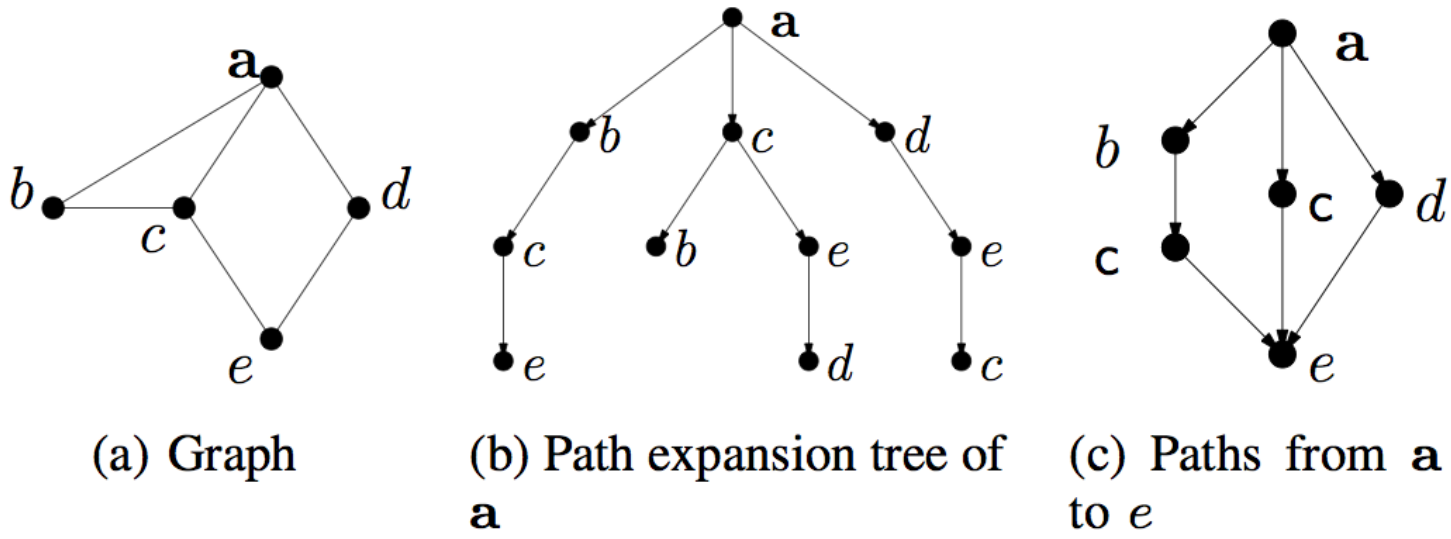
Meaningful Influence Path in IC model

$$p = \langle v_1, v_2, \cdots, v_m \rangle$$

$$ipp(p) = \begin{cases} 0 & , p = \langle \rangle \\ \prod_{i=1}^{m-1} w_{(v_i, v_{i+1})} & , p = \langle v_1, \cdots, v_m \rangle \end{cases}$$

$$ipp(p) \geq \theta$$

Gathering influence paths



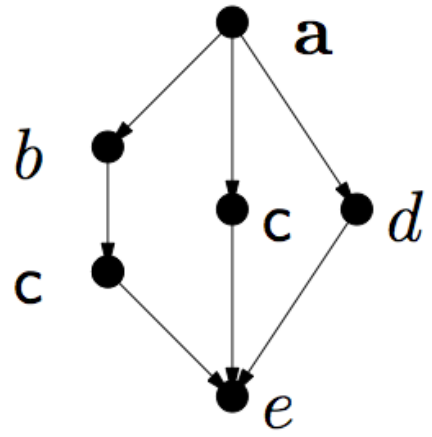
(a) Graph (b) Path expansion tree of **a** (c) Paths from **a** to e

Easily obtained by graph traversal

Approximating σ({v})

$$\hat{\sigma}(\{v\}) = 1 + \sum_{u \in O_v} \hat{\sigma}_u(\{v\})$$

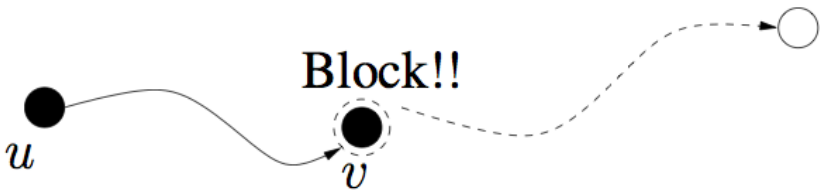
$$\hat{\sigma}_u(\{v\}) = 1 - \prod_{p \in P_{v \to u}} (1 - ipp(p))$$



$\hat{\sigma}_e(\{\mathbf{a}\}) =$
$1 - \{(1 - 0.0001)(1 - 0.001)(1 - 0.001)\} =$
0.0021

Approximating σ(S ∪ {v}) - σ(S)

- σ({v}) ≠ σ(S ∪ {v}) - σ(S)
  - influence blocking!!!!



- We should detect blocked(invalid) paths

Detecting influence blocking

- Current seed set : S

$$\langle \cdots, v, \cdots \rangle \in P_{S \to V}$$

- New seed node : v

$$\forall u \in S.(\langle v, \cdots, u, \cdots \rangle \in P_{v \to V})$$

- Valid Paths

$$P_{S \to u}^{valid} = \{p | p \in P_{S \to u}, |p \cap S| = 1\}$$

Approximating $\sigma(S \cup \{v\}) - \sigma(S)$

$$\hat{\sigma}(S \cup \{v\}) - \hat{\sigma}(S) = 1 + \sum_{u \in O_v \cup \{v\}} \{\hat{\sigma}_u(S \cup \{v\}) - \hat{\sigma}_u(S)\}$$

$$\hat{\sigma}_u(S) = 1 - \prod_{p \in P_{S \to u}^{valid}} (1 - ipp(p))$$

CT-IPA : an extension of IPA

- All we need is redefining ***ipp(p)!!***

$$p = \langle u = u_0, u_1, \cdots, u_{l-1}, u_l = v \rangle$$

$$ap(v,t) = \sum_{i=0}^{t-1} c_{uv}^{(t-i)} ap(u,i)$$

$$= \begin{bmatrix} ap(u,0) \\ ap(u,1) \\ \vdots \\ ap(u,t-1) \end{bmatrix}^{\mathrm{Tr}} \begin{bmatrix} c_{uv}^{(t)} \\ c_{uv}^{(t-1)} \\ \vdots \\ c_{uv}^{(1)} \end{bmatrix},$$

$$c_{uv}^{(t-i)} = pp_{t-i-1}(u,v) \prod_{j=0}^{t-i-2} (1 - pp_j(u,v))$$

CT-IPA : an extension of IPA

- In a matrix form

$$\begin{bmatrix} ap(v,0) \\ ap(v,1) \\ ap(v,2) \\ \vdots \\ ap(v,T) \end{bmatrix}^{\mathrm{Tr}} = \begin{bmatrix} ap(u,0) \\ ap(u,1) \\ ap(u,2) \\ \vdots \\ ap(u,T) \end{bmatrix}^{\mathrm{Tr}} \begin{bmatrix} 0 & c_{uv}^{(1)} & \cdots & c_{uv}^{(T)} \\ 0 & 0 & \cdots & c_{uv}^{(T-1)} \\ 0 & 0 & \cdots & c_{uv}^{(T-2)} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$\mathbf{AP}(v) = \mathbf{AP}(u)\mathbf{C}_{uv}$$

CT-IPA : an extension of IPA

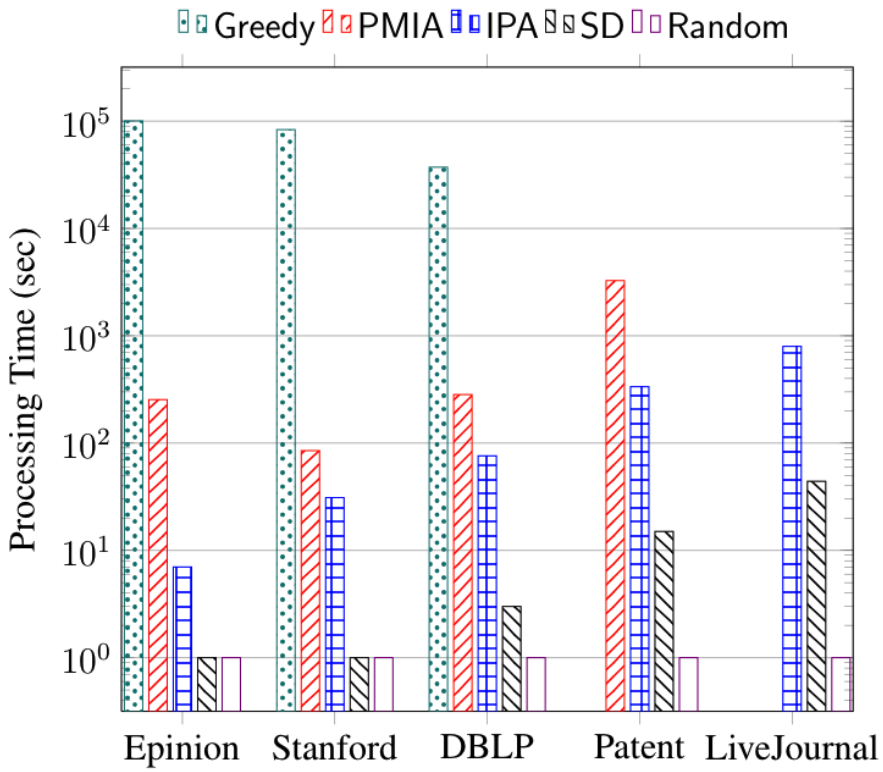
- Finally

Lemma 3: The probability that $u \in S$ activates $v \in V \backslash S$ only through a path $p = \langle u = u_0, u_1, \cdots, u_{l-1}, u_l = v \rangle$ is

$$ipp(p) = [1\ 0\ \cdots\ 0] \left(\prod_{i=0}^{l-1} \mathbf{C}_{u_i u_{i+1}} \right) [1\ 1\ \cdots\ 1]^{\mathrm{Tr}}, \quad (2)$$

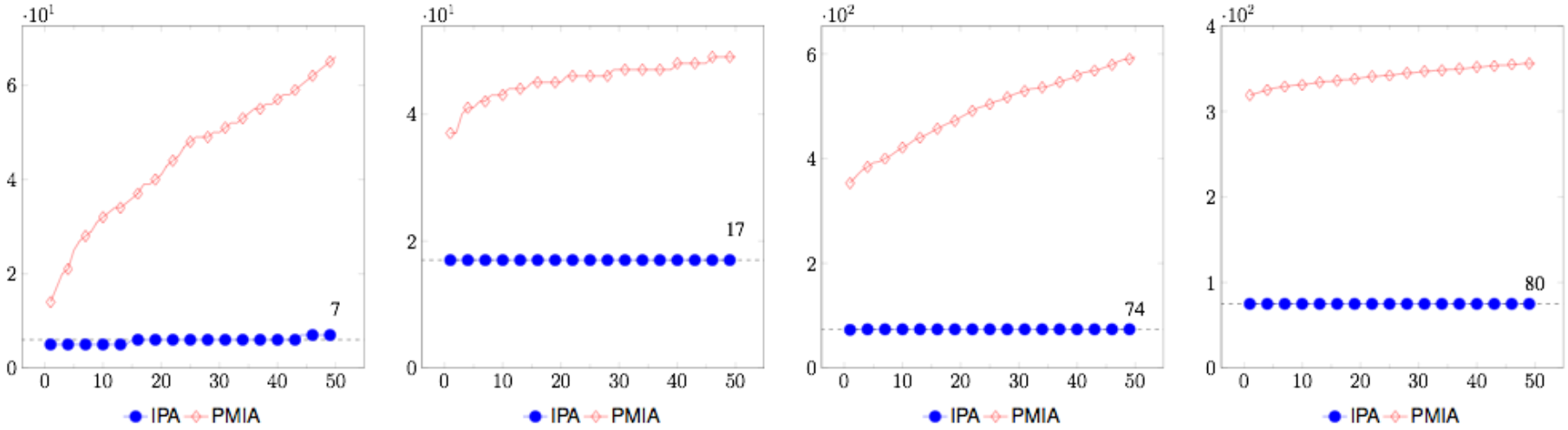
where $u_i \in V \backslash S$ for all $i = 1, \cdots, l$, and the order of matrix multiplication is from $i = 0$ to $l - 1$.

Dataset	Epinion	Stanford	DBLP	Patent	LiveJournal
# of Nodes	75.8K	281K	655K	3.77M	4.85M
# of Edges	509K	2.31M	3.98M	16.5M	69.0M
Average Degree	6.71	8.20	6.10	4.38	14.2
Max In/Out Degree	3032/1798	38606/255	588/588	779/770	13906/20293
Direction	directed	directed	undirected	directed	directed



Processing Time

Progressiveness



(a) Epinion (b) Stanford (c) DBLP (d) Patent

Fig. 5. Processing time as the number of seed nodes increases (X axis : number of seed nodes, Y axis : processing time in seconds)

Memory Handling

TABLE III

PRIORITY QUEUE SIZE VS. INFLUENCE SPREAD ($k = 50$)

priority queue size	Epinion	Stanford	DBLP	Patent	LiveJournal
k	12496	21172	7533	10551	106745
$2k$	12486	25505	7549	10899	107469
$3k$	12476	25509	7550	10902	107482
$4k$	12482	25509	7553	10900	107451

TABLE IV

MEMORY USAGE OF IPA ($k = 50$)

priority queue size	Epinion	Stanford	DBLP	Patent	LiveJournal
IPA($\|V\|$)	$1.6GB$	$9.2GB$	$17GB$	$\times(> 24GB)$	$\times(> 24GB)$
IPA($3k$)	**207MB**	**597MB**	**419MB**	**1.6GB**	**3.6GB**
PMIA($\|V\|$)	$418MB$	$1.6GB$	$5.5GB$	$16GB$	$\times(> 24GB)$

Influence

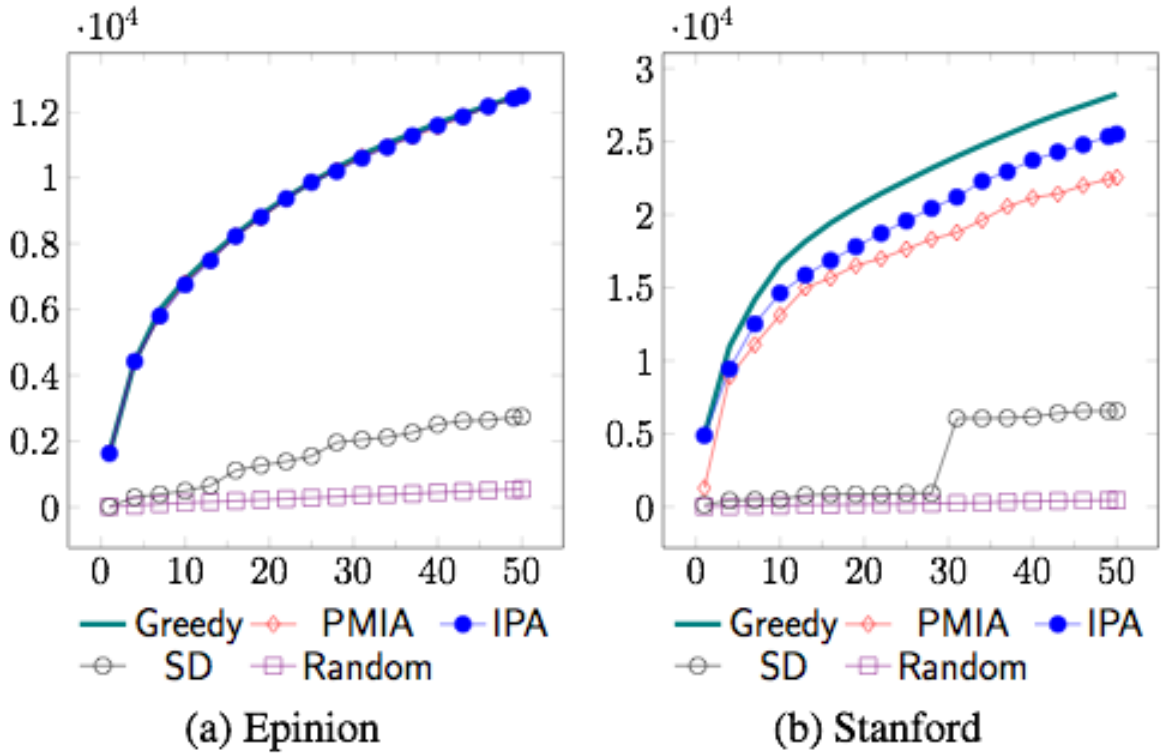


Figure 5: Influence Spread in Epinion and Stanford
(X axis : number of seed nodes, Y axis : influence spread)

Influence

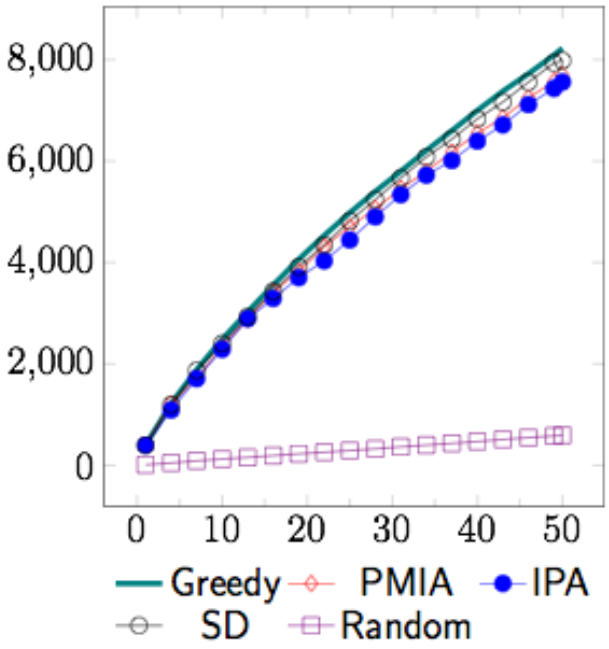


Figure 6: Influence Spread in DBLP
(X axis : number of seed nodes, Y axis : influence spread)

Influence

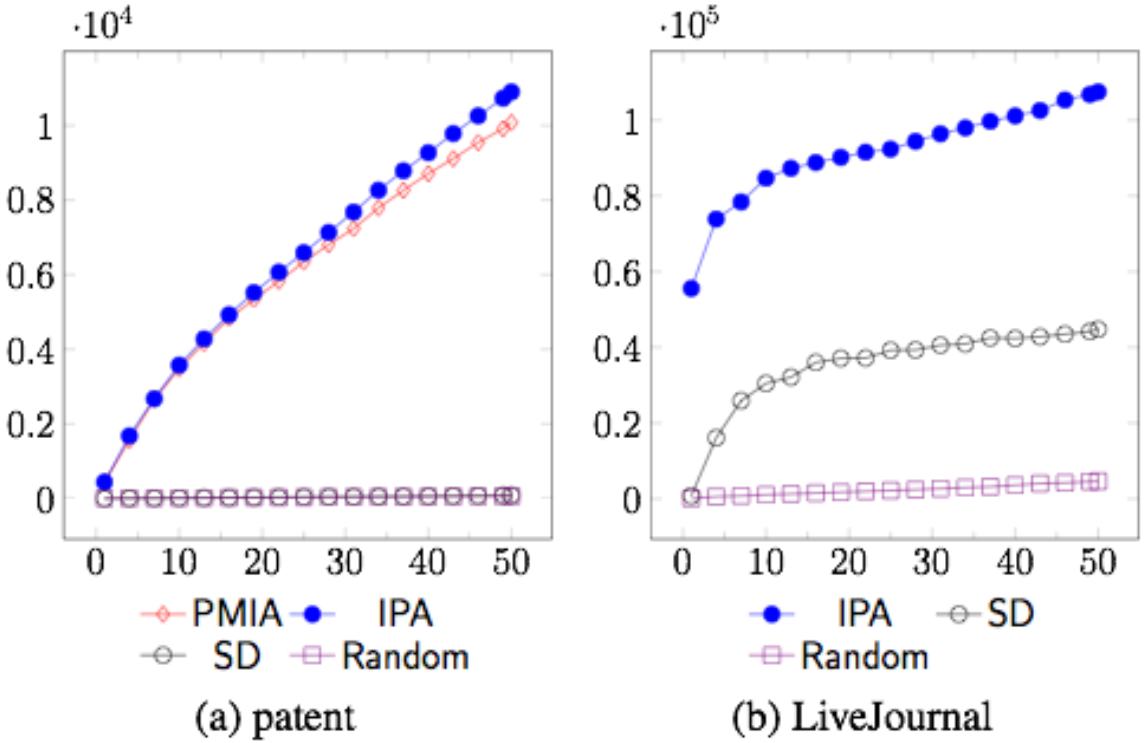
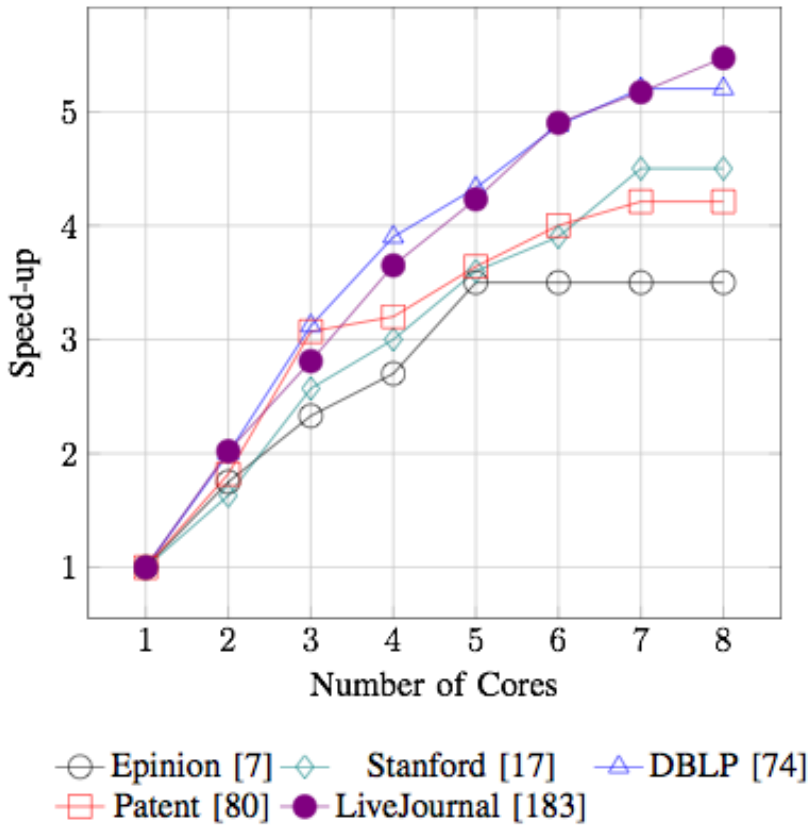


Figure 7: Influence Spread in Patent and LiveJournal
(X axis : number of seed nodes, Y axis : influence spread)

Parallelization Effect



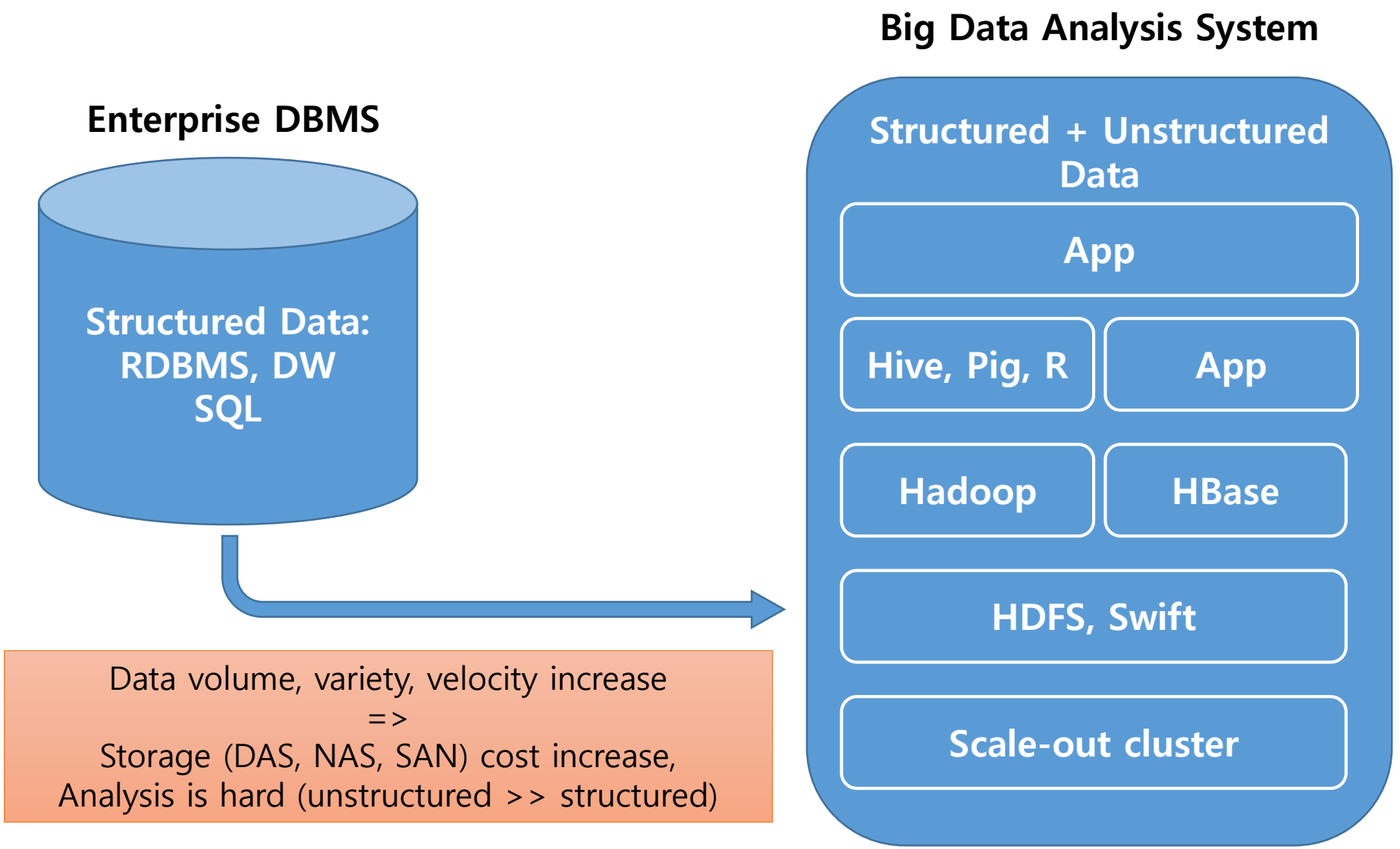
Scaling up to Billion-Nodes Network using Map-Reduce?

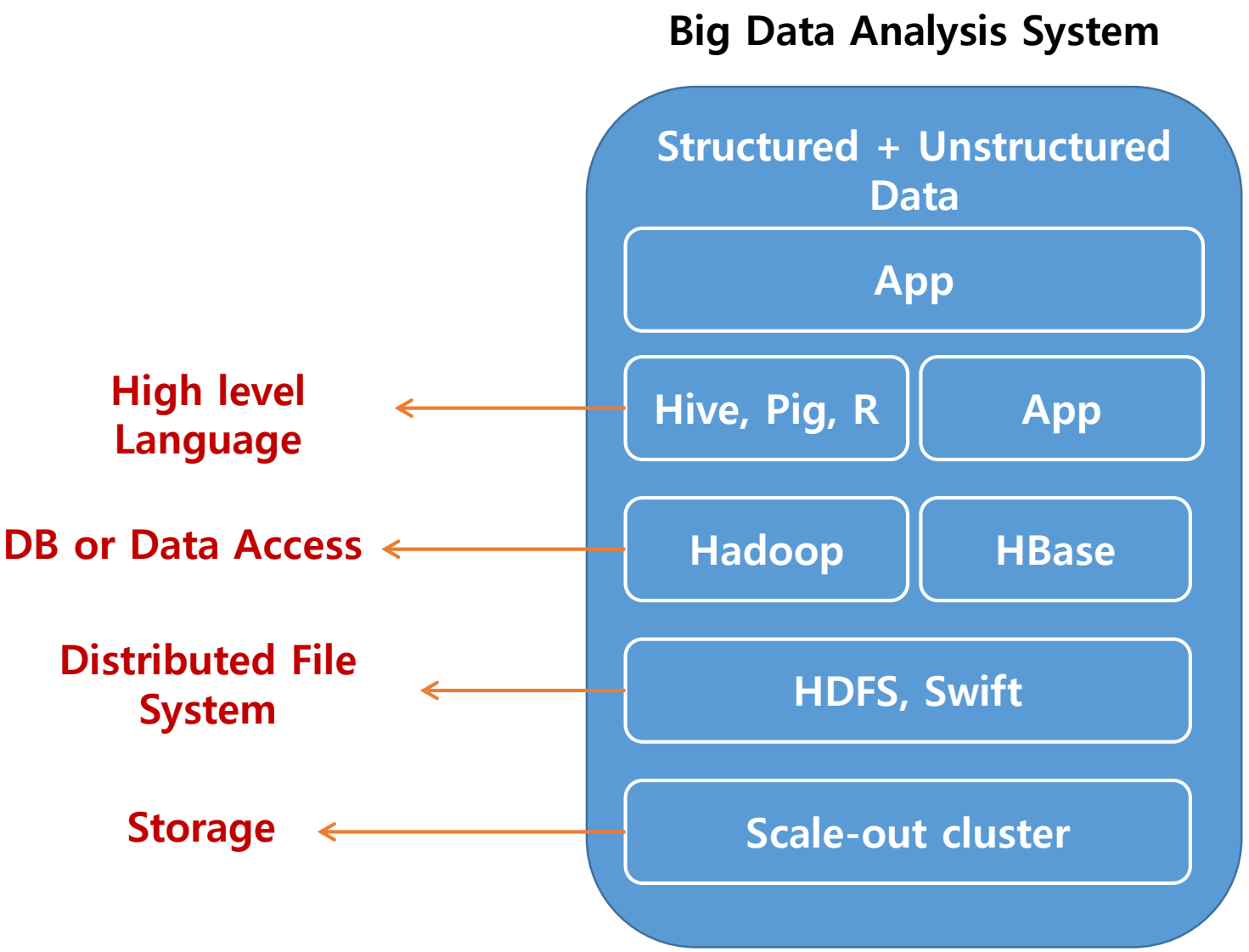
Very Hard !

Something is easily parallelized does NOT mean it can be easily "map-reduced".

Big data processing ≠ Parallel data processing

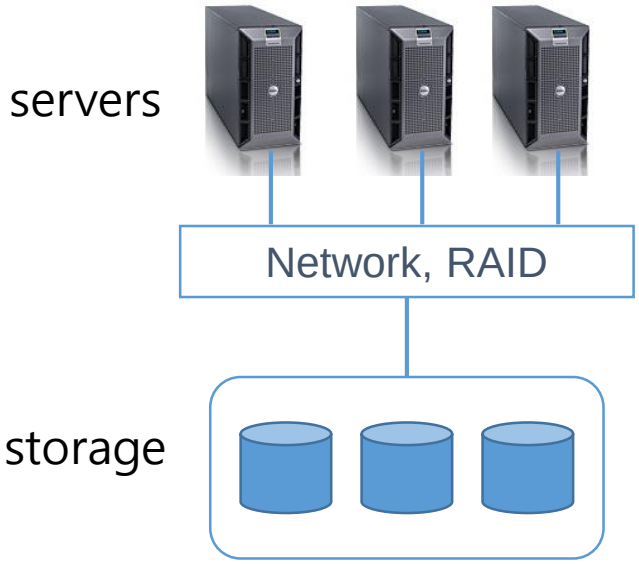
How different?


Big Data Analysis System
Enterprise DBMS
Structured Data:
RDBMS, DW
SQL
Structured + Unstructured Data
App
Hive, Pig, R
App
Hadoop
HBase
HDFS, Swift
Scale-out cluster
Data volume, variety, velocity increase
=>
Storage (DAS, NAS, SAN) cost increase,
Analysis is hard (unstructured >> structured)


Big Data Analysis System
Structured + Unstructured Data
App
High level Language
Hive, Pig, R
App
DB or Data Access
Hadoop
HBase
Distributed File System
HDFS, Swift
Storage
Scale-out cluster

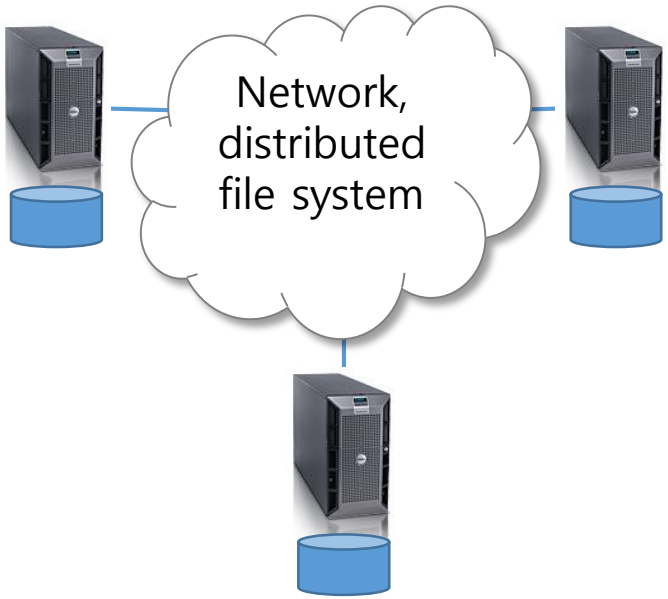
Storage Trend

Centralized storage: SAN, NAS



Big data =>
Need scalability

Distributed storage



- **Proprietary, Highly reliable HW**

=> Scale-up: Expensive

=> Fast data transfer

- **Commodity HW**

=> Scale-out: Inexpensive

=> Slow data transfer

=> Need new programming model !

Trend Evolution

Data Trend (Big Data)

Storage Trend (Distributed) : Inexpensive Scale-out, but
Expensive Data Transfer!

Need New Programming Model to Minimize Data Transfer

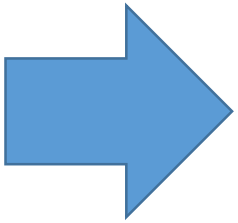
Move operations instead of data!

MapReduce by Google

Hadoop and many subprojects

MapReduce Principles

- Run operation on data nodes: Move operations to Data
- Minimize data transfer



Design Tips

- Lower the work of reduce
  - Use combine if possible
- Compression of map's output helps decreasing network overhead
- Minimize iterations and broadcasting
  - Sharing information is minimized
- Use bulk reading
  - Too many invocation of map may incur too many function calls
- Design algorithm to have enough reduce functions
  - Having only a single reduce will not speed up

Programming is Hard!!!

A straightforward extension of parallel IPA algorithm produce *too many iterations* and *heavy data transfer from map to reduce*

Big Data Subprojects

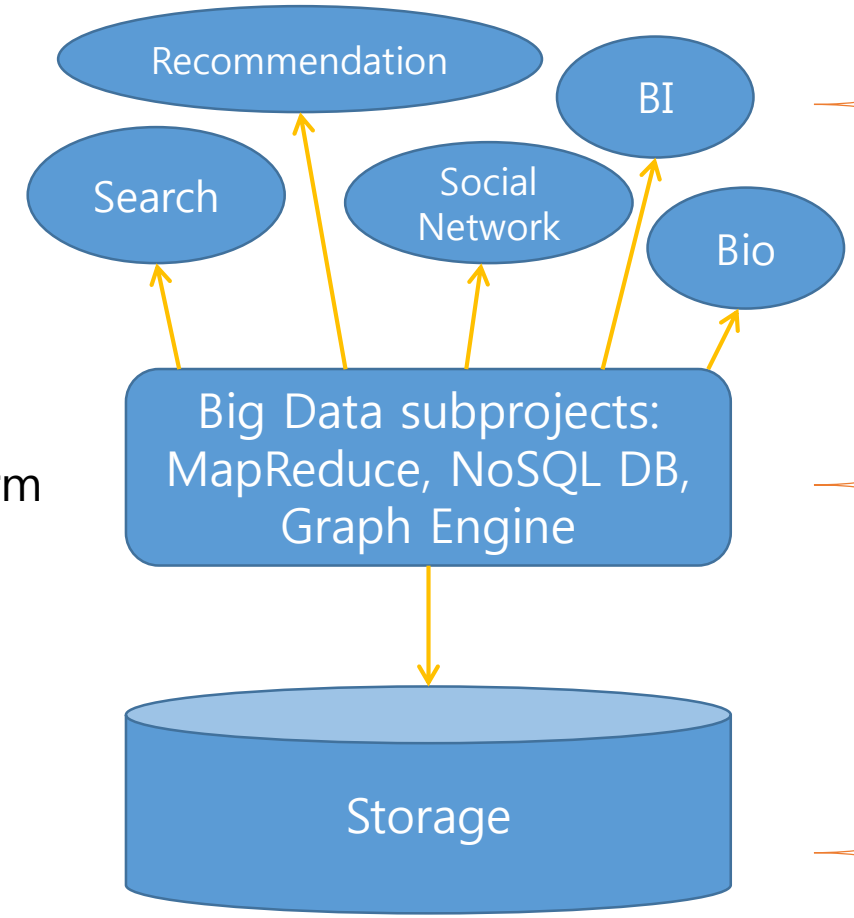
- Big data programming framework
  - MapReduce (Batch): HDFS & Hadoop, Dryad
  - MapReduce (Iterative): HaLoop, Twister
  - MapReduce (Streaming): Storm (Twitter), S4 (Yahoo), InfoSphere Streams (IBM), HStreaming
- NoSQL DB
  - HBase (Master, slaves), Cassandra (P2P, "Gossip", no master server), Dynamo (Amazon), MongoDB (for text), …
- Graph processing engine
  - Pregel, Giraph, Trinity, Neo4J

Big Data Solutions

- **Open source solutions >> Closed solutions**
  - Commercial systems such as EMC and Oracle also use open sources like Hadoop and Hbase.

- Big data systems are composed of ***Hadoop*** and many related ***subprojects***

- Each subproject has its own ***characteristics*** and ***functions*** => require much ***experience*** and ***know-how*** to understand and efficiently handle them to develop a Big data processing system



App

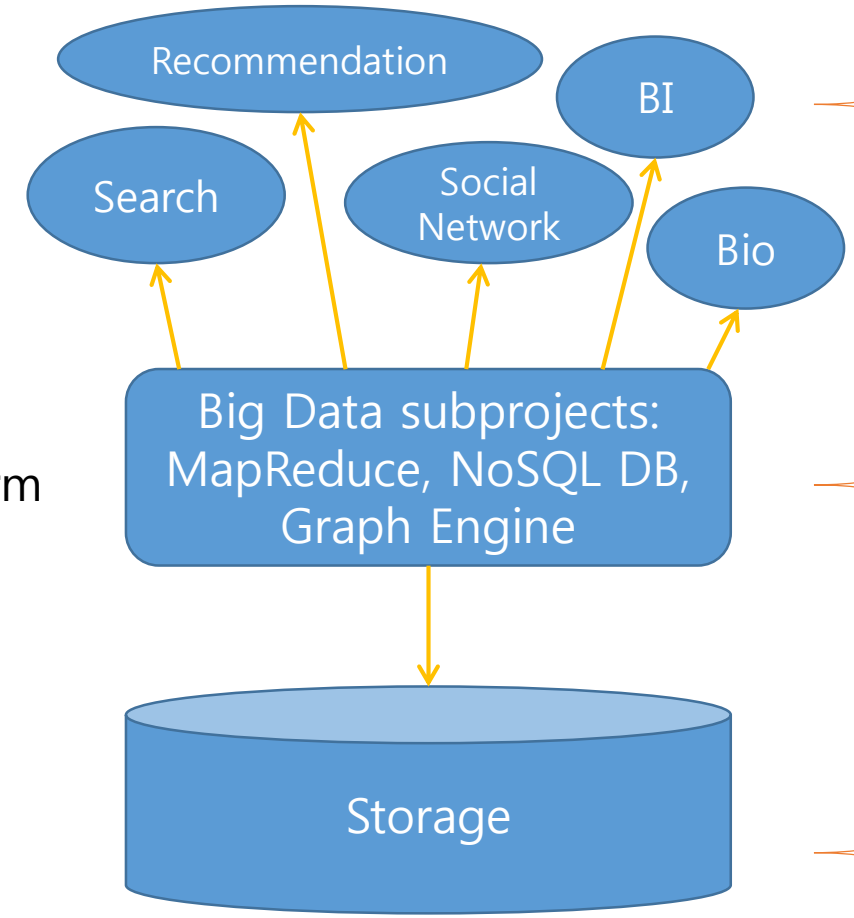
SW Platform

HW Intra

- Minimize Data Transfer
- Which platform?
- Generalization
- Feasible? Approximate?
- SSD-aware mining

- Move CPU to Data
- Minimize Data Transfer
- Search, Recommendation, ..
- Text, Graph, Multimedia, ..
- Batch, Streaming
- SSD-aware platform

- Scalability
- Scale-out cost
- Energy efficiency
- Load balancing
- SSD where?



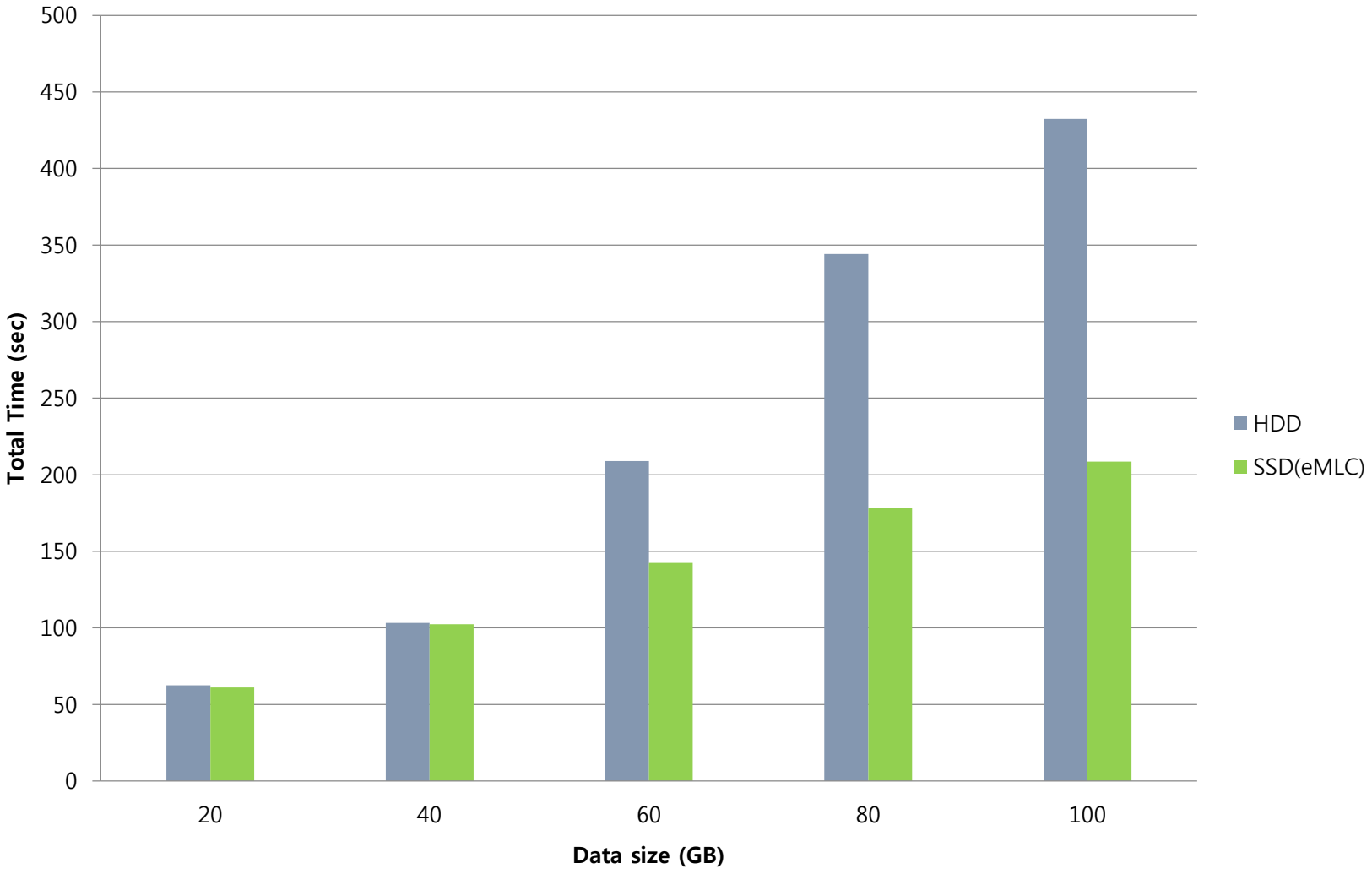
- Minimize Data Transfer
- Which platform?
- Generalization
- Feasible? Approximate?
- **SSD-aware mining**

- Move CPU to Data
- Minimize Data Transfer
- Search, Recommendation, ..
- Text, Graph, Multimedia, ..
- Batch, Streaming
- **SSD-aware platform**

- Scalability
- Scale-out cost
- Energy efficiency
- Load balancing
- **SSD where?**

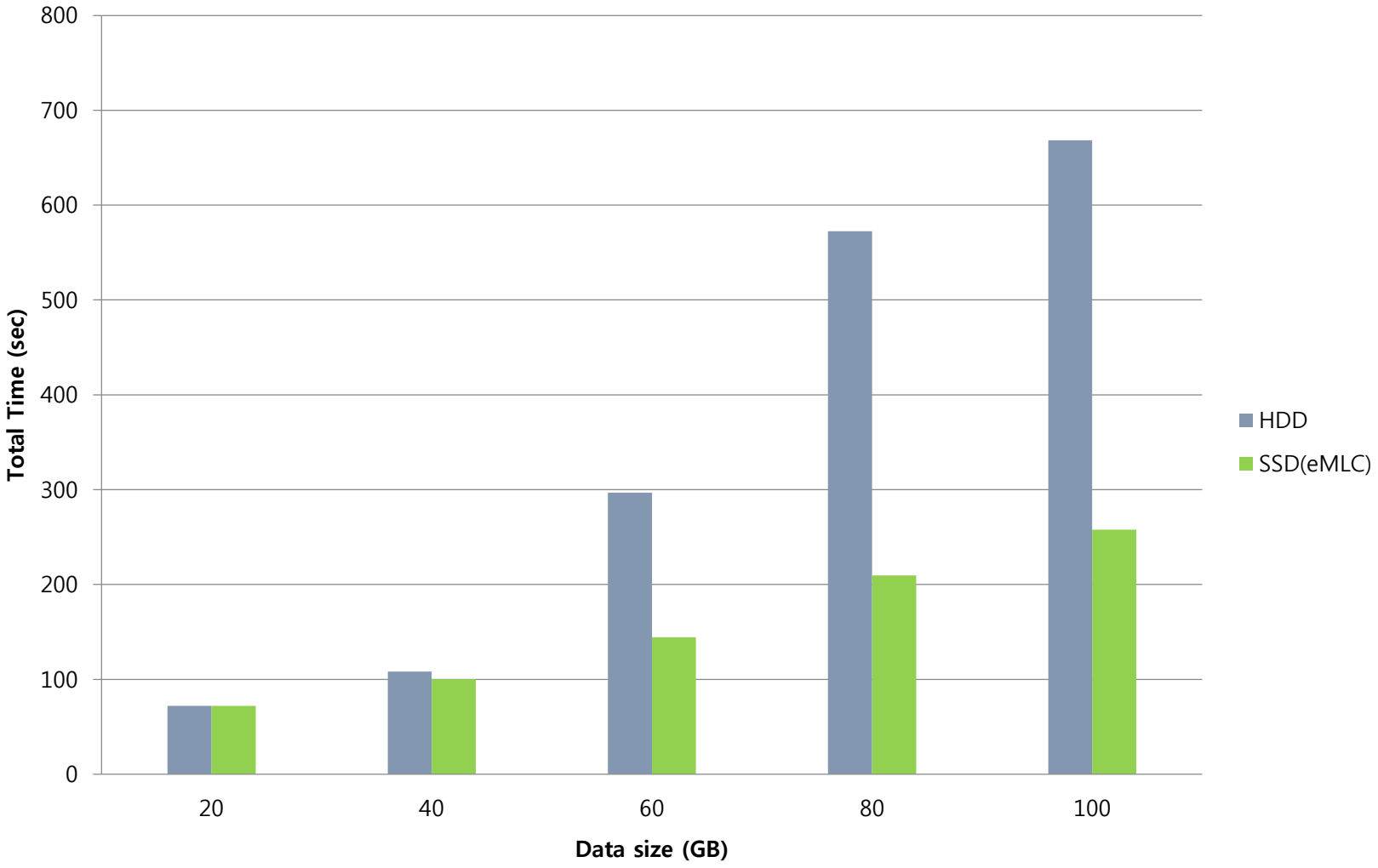
Teragen: HDD (7200 RPM) vs SSD (eMLC)

(16 nodes, 160 mappers)

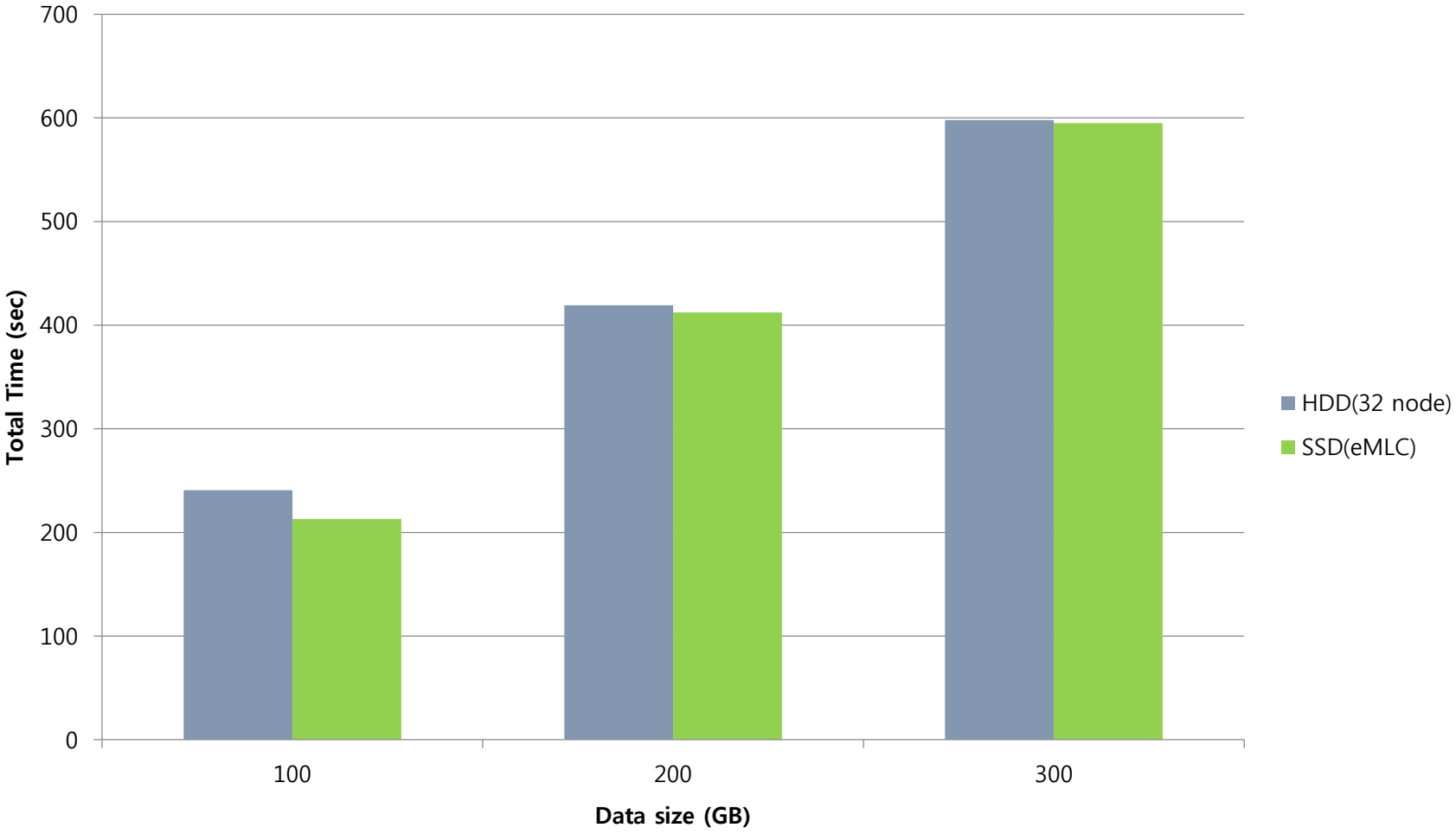


Terasort: HDD (7200 RPM) vs SSD (eMLC)

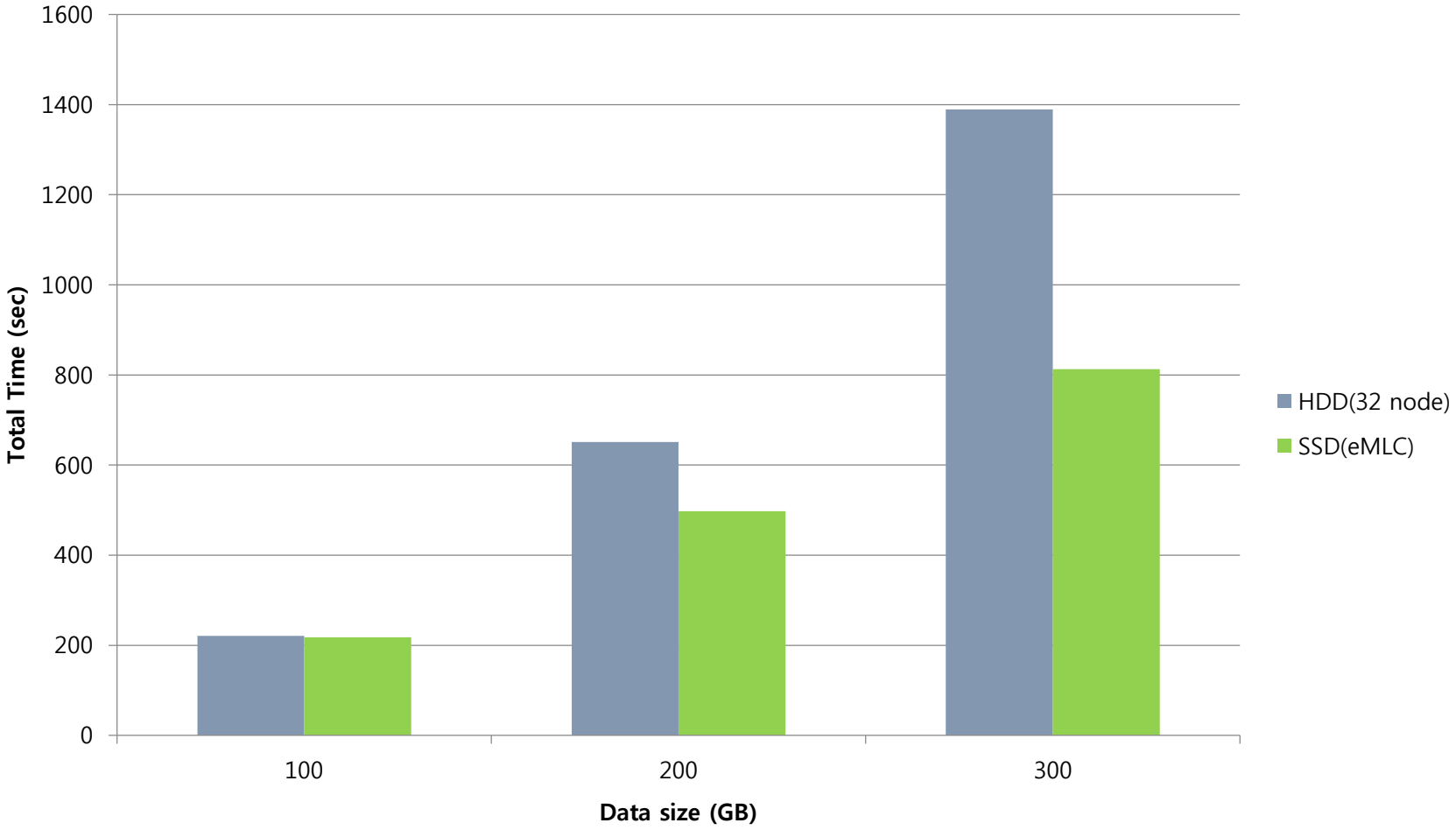
(16 nodes, 160 mappers)



Teragen: HDD (32 nodes) vs SSD (16 nodes)

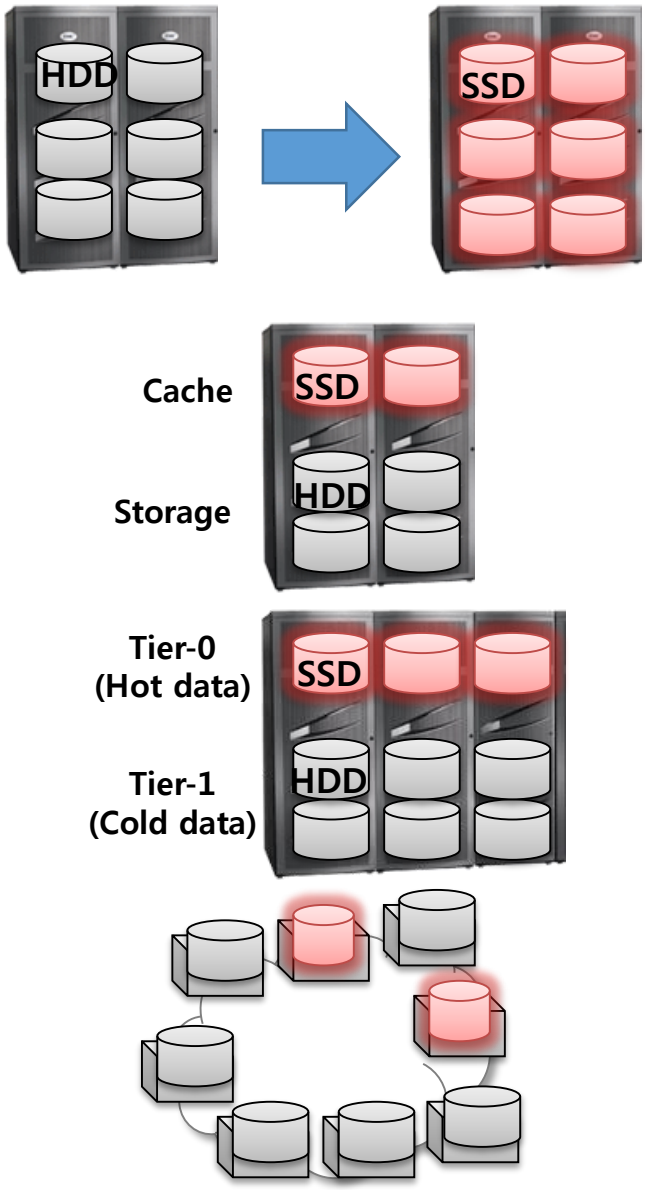


Terasort: HDD (32 nodes) vs SSD (16 nodes)



SSD where?

- Replacement model
  - Replace HDD with SSD
  - Throw out HDD?
  - Big data => Expensive scale-out?
- Caching model
  - Use SSD as cache between memory and HDD
  - Ratio of SSD and HDD?
  - Data duplication?
- Tiering model
  - Put hot data to SSD and cold data to HDD
  - Data migration?
- **Distributed model**
  - **Don't care migration, don't care ratio, no duplication, no need to throw out HDDs**
  - **Load balancing by Hadoop**



Reality

- Linkedin
  - Develop NoSQL database "Voldemort" which uses SSDs
  - Twitter
    - Optimize MySQL for SSDs (e.g., page-flushing behavior, reduction in writes to disk)
- Amazon
  - Develop SSD-based NoSQL database "DynamoDB" as a new service in AWS
- EBay
  - Replace its internal virtual storage layer with 100TB SSDs (2011)
  - Saw 50% reduction in rackspace, 78% drop in power consumption and a 5 times boost in I/O performance.
- Microsoft
  - Replace Bing Search runtime filesystem with Intel SSD (2011)
  - Uses Intel SSDs in their KeyValue storage for Bing social search
  - Microsoft Research is working on Flash Server Farm Called CORFU (Cluster Of Raw Flash Units)
- Facebook
  - Improve MySQL performance by adding Fusion-io as caching layer

Reality

Replacement models > Caching models >> Distributed Model

Develop own solutions > Use Big data subprojects

SSD and Hadoop

Most companies use the replacement or caching models.

Distributed model is promising

but

No SW platform are yet developed for it.

SSD Research for Big Data Processing

Data transfer between nodes is expensive.

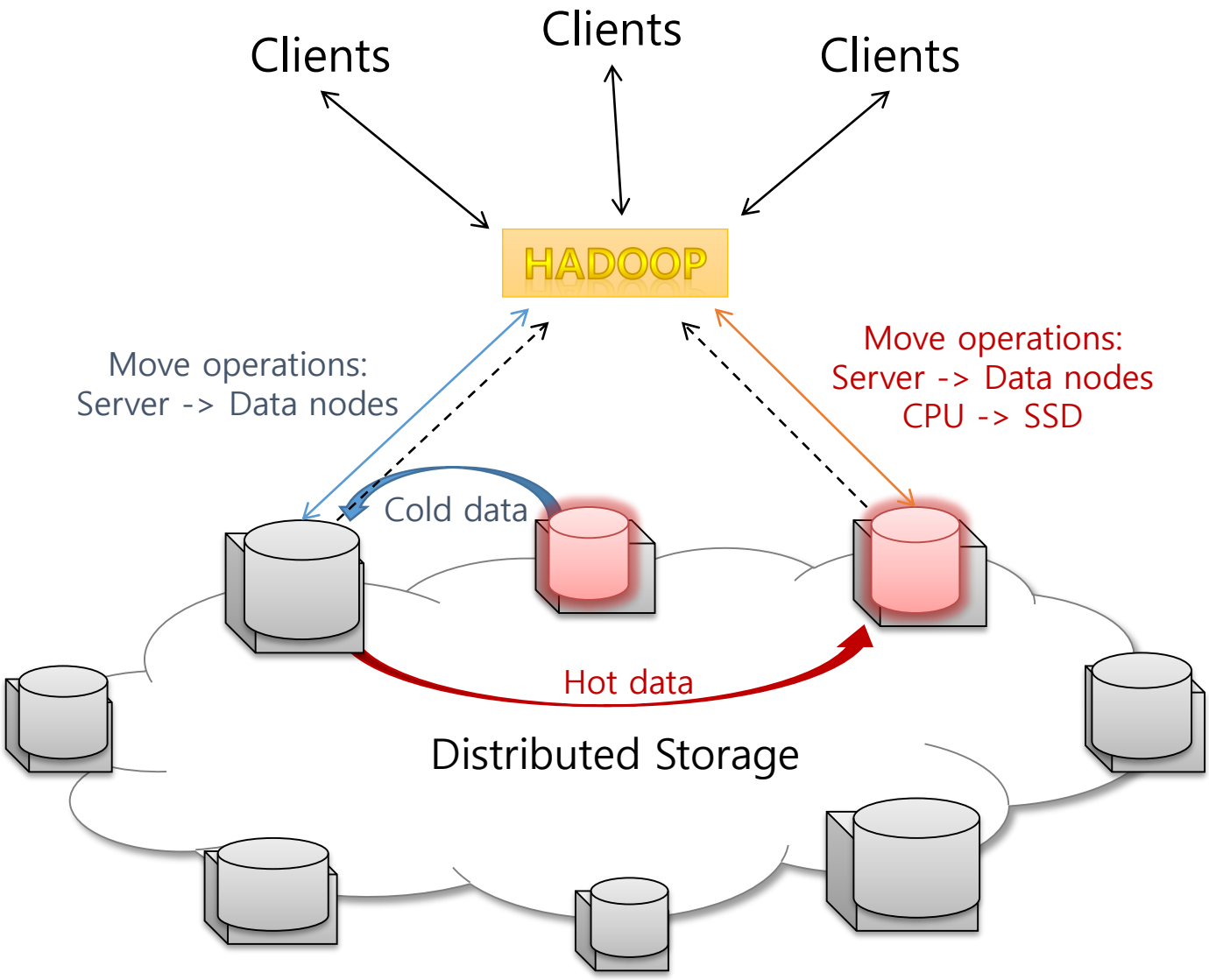
Hadoop: Move operations from server to data nodes

(Macro-level trend)

Data transfer between CPU and SSD is expensive.

SSD: Move operations from CPU to SSD ?

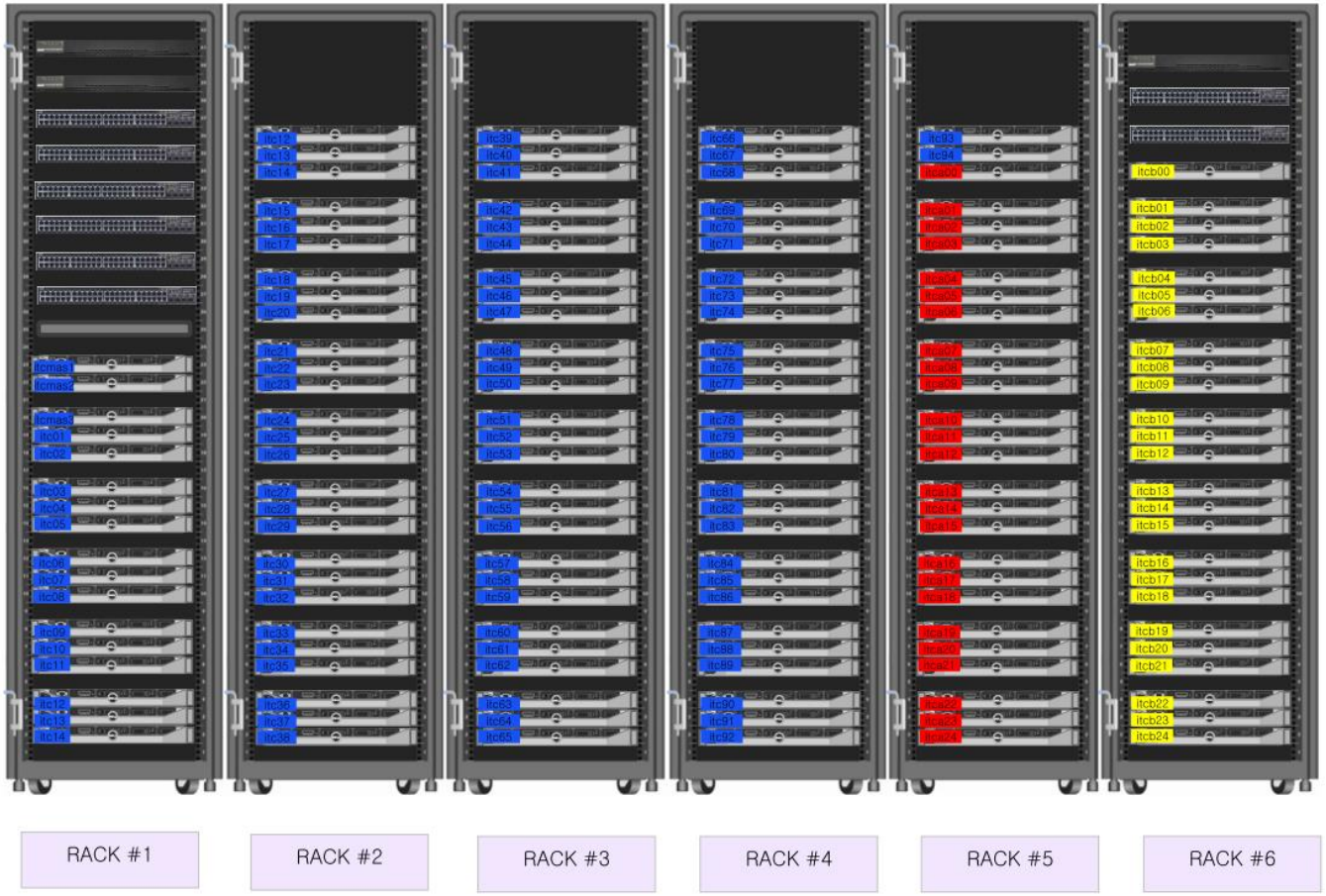
(Micro-level trend ?)


Clients
Clients
Clients
HADOOP
Move operations:
Server -> Data nodes
Move operations:
Server -> Data nodes
CPU -> SSD
Cold data
Hot data
Distributed Storage

Q/A

POSTECH Cluster for big data analytics

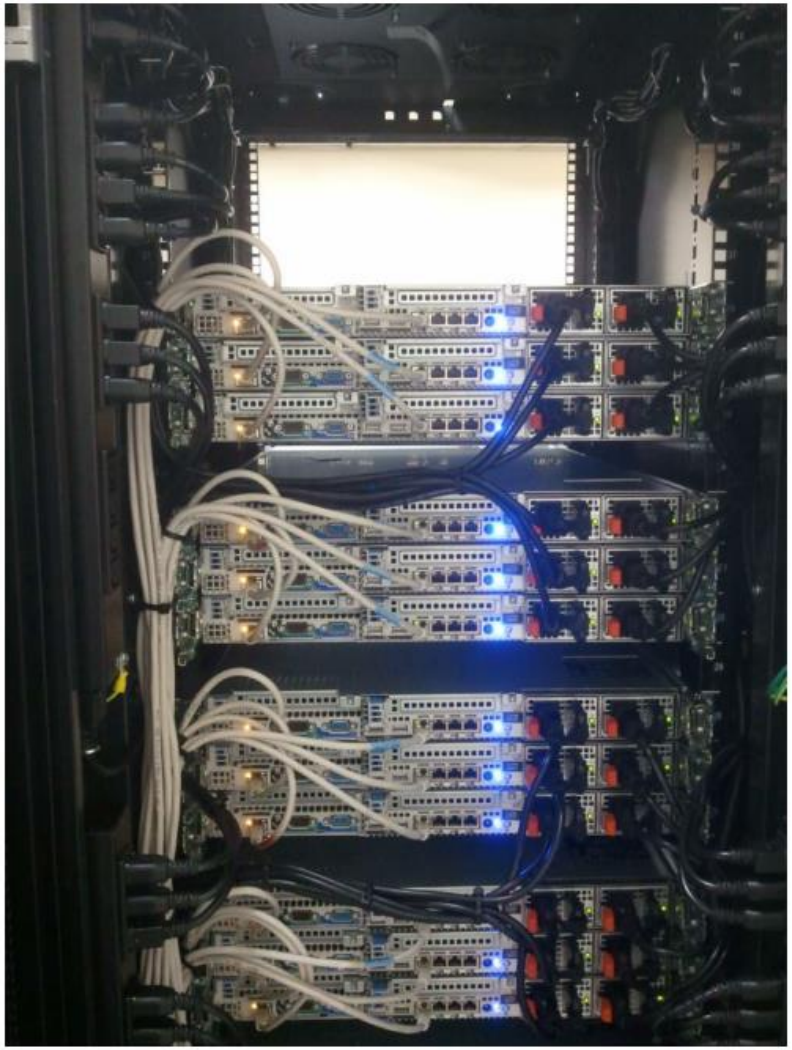
POSTECH Cluster System - Architecture



POSTECH Cluster System - Architecture

POSTECH Cluster System - Architecture

POSTECH Cluster System - Architecture



POSTECH Cluster System (150 nodes)

- Hardware
  - DELL PowerEdge R610
  - 150 nodes (100 for data mining research, 50 for others)
  - 2 G NET (2 * 1 G)
- Each node
  - 12 cores (2 * six-core CPU: Intel Xeon X5650 2.66GHz)
  - 24 Gb memory (6 * 4Gb: 1333Mhz Dual Ranked RDIMMs)
  - 3 Tb HD (6 * 500Gb 7200k rpm SATA)
- Software
  - CentOS 5.5 (x86_64)
  - Intel Compiler, MPI, SGE, Hadoop
- **Purchased another 150 SSD nodes, each 16 cores, 36G mem, 4T HDD, 200G SSD!**

DM lab. at POSTECH

- Projects
  - Big Data Mining (with 교과부)
  - Mobile Data Mining and Search (with 교과부)
  - Energy Efficient Mining for Mobile Devices (with 지경부)
  - Big Data Mining with SSD (with …)
  - Open Data Market (with …)
  - Mining for Online Advertisers (with …)

- Recent Publications
  - Recommendation [ICDM 2011]
  - Mining for Online Advertisers [CIKM 2011]
  - Location Privacy on Mobile Devices [KDD 2011]
  - Social Network Mining [ICDM 2012, ICDE 2012]
  - Mobile Multimedia Search [KDD 2012]
  - Relevance Feedback Search [SIGMOD 2011, KDD 2012]

Q/A