CREST 戦略的創造研究推進事業
Core Research for Evolutional Science and Technology

JST Japan Science and Technology Agency

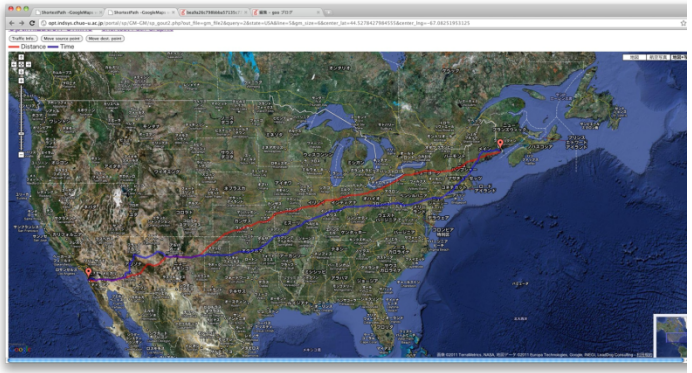
スーパーコンピュータを用いた大規模グラフ解析とGraph500 ベンチマーク

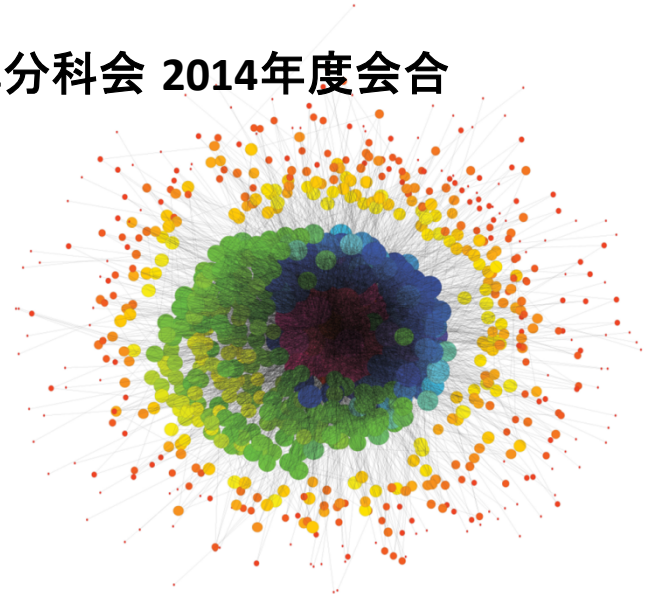
藤澤 克樹
九州大学マス・フォア・インダストリ研究所＆ JST CREST

共同研究者：上野 晃司（東工大), 安井 雄一郎(九大), 鈴村 豊太郎（University College Dublin), 佐藤 仁, 岩渕 圭太, 溝手 竜(東工大)

2014年10月29日

サイエンティフィック・システム研究会 科学技術計算分科会 2014年度会合
次世代HPCを支える技術
ホーテルオークラ神戸





当 CREST の研究計画の概要

- 次世代ポストペタスパコンでの解決すべき課題の特定
  - 並列数の爆発的増大、不均質化、高密度化
  - 記憶装置の多階層化・大容量化
  - アルゴリズム的、システム的に様々な解決すべき課題と困難が存在
- **大規模グラフ解析及び数理最適化システム**
  - 緊急に取り組むべき課題と実社会へのインパクト
  - Graph500(Green Graph500)ベンチマーク (巨大グラフ, BFS)
    - ISC12 : 358GTEPS (世界3位) , 8.15GTEPS (1ノード世界1位)
    - SC12 : 5524GTEPS(世界4位), 10.495GTEPS(1ノード世界1位)
    - ISC13 : 6.119GTEPS/kW (世界1位)
    - SC13 : 6.72GTEPS/kW (BigData 1位), 153.17GTEPS/kW (SmallData 1位)
    - ISC14 : 17977GTEPS(世界1位), 59.12GTEPS/kW (BigData 1位)
  - 数理計画問題(SDP): (世界記録更新:233万制約 ; 1.713PFlops)
    - SC12(Tech. paper) , IPDPS14 など: 疎&密データ計算(24,480CPUコア & 4080GPU) : 日本オペレーションズリサーチ学会研究賞、NVIDIA GTC Japan 最優秀ポスター発表賞等
- 最適化とHPC系研究者のポストペタスパコン上での **Co-design による解決**
- ポストペタスパコン上での基盤ソフトの整備に貢献すると共に安心安全な社会の実現を目指す

超大規模グラフ最適化システムの提案

防災計画策定

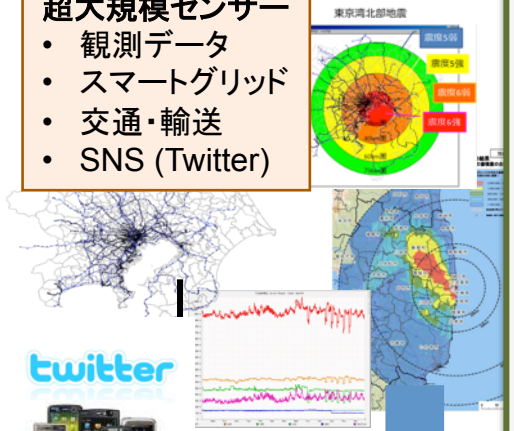
交通・災害復興・
避難・ロジスティクス

エネルギー・省電力

ソーシャル
ネットワーク解析

超大規模グラフ最適化システム

超大規模センサー
- 観測データ
- スマートグリッド
- 交通・輸送
- SNS (Twitter)



データソース

データソース

大規模グラフ可視化

リアルタイム大規模グラフ
ストリーム処理

グラフ探索及び数理最適化
ライブラリによる大規模グ
ラフ処理

中心性
解析

最短経路
問題

最速
フロー問題

Indexing

クラスタ
リング

半正定値
計画問題

混合整数
計画問題

リアルタイム
ストリーム処理系

X10言語処理系

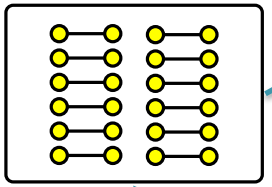
100PF級ヘテロジニアススパコン

大規模グラフストア

応用分野

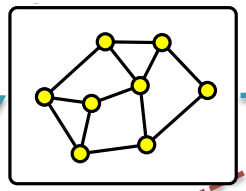
交通ネットワーク
ソーシャルネットワーク
サイバーセキュリティ
バイオインフォマティクス
脳科学
防災計画策定

事象同士の関係
(Relationships)



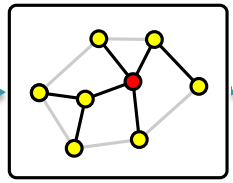
Step1

グラフ



Step2
グラフ解析

解析結果



Step3
分析と理解

Twitter

ソーシャルネットワーク
6,160万点　&　14億7千万枝

- 並列グラフ探索 (幅優先探索)
- 最適化 (最短路, 最大フロー, 最小費用フロー)
- クラスタリング (グラフ分割, コミュニティ抽出)

ニューラル・ネットワーク @ Human Brain Project
890億点 & 100 兆枝

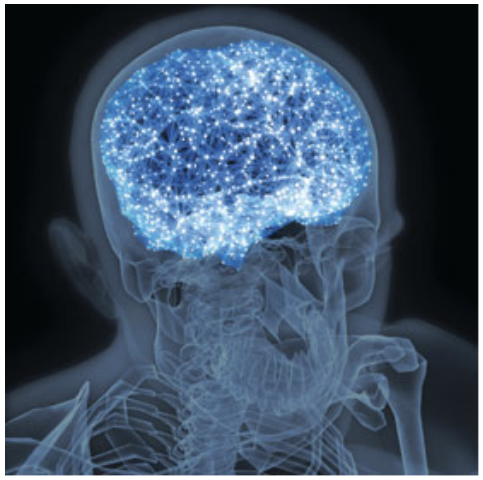
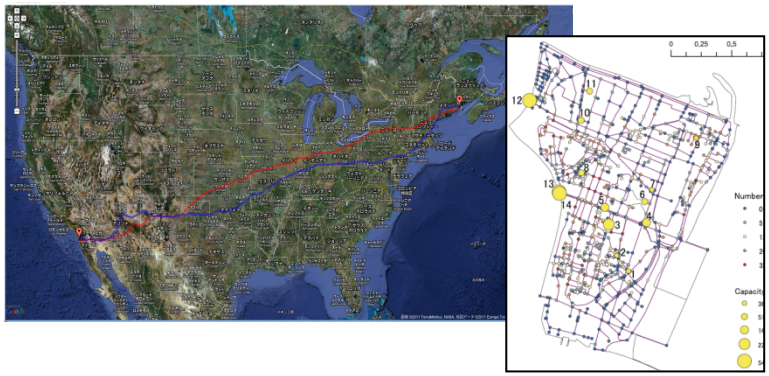


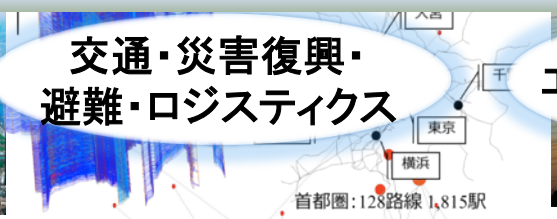
Image: Illustration by Mirko Ilic

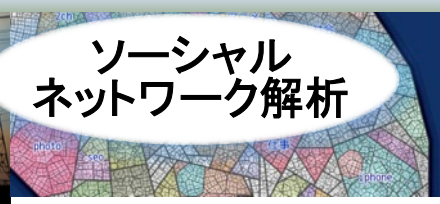
全米道路ネットワーク
2,400万点 & 5800万枝

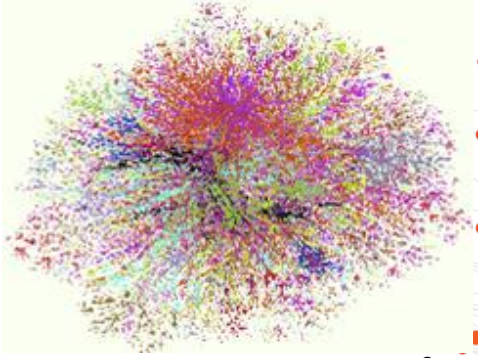


サイバーセキュリティ
150億/日 のアクセスログ

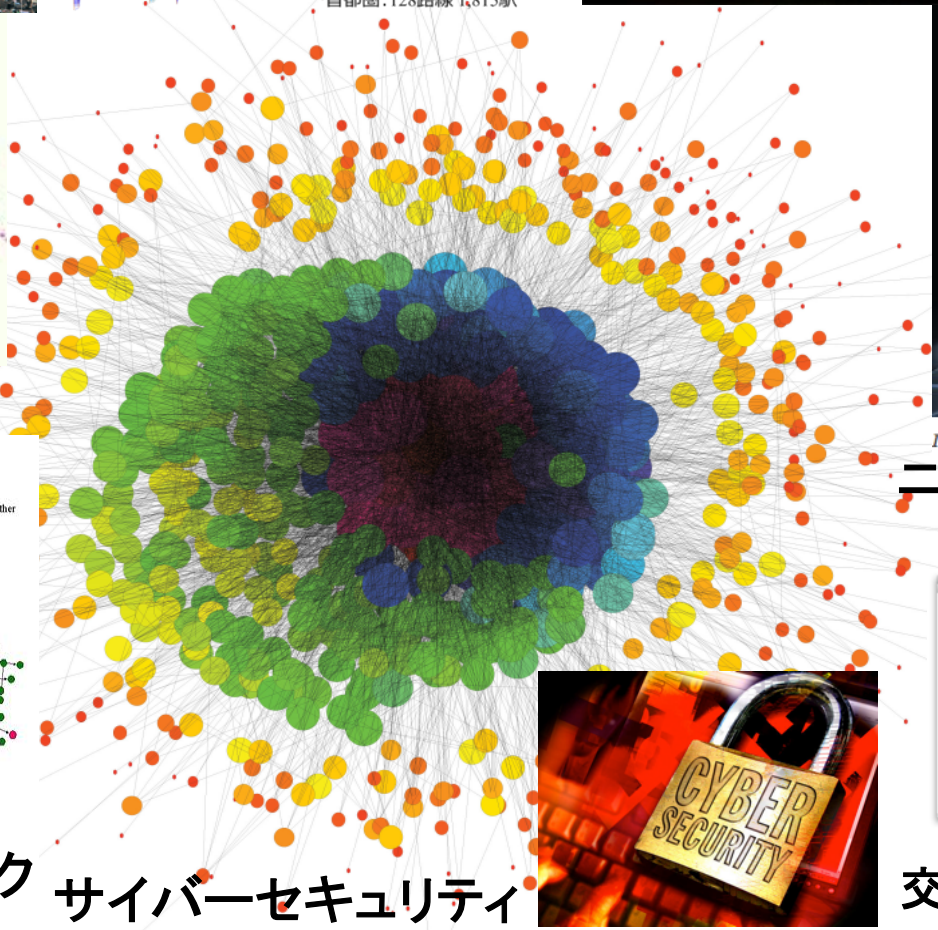
大規模グラフ解析：緊急に取り組むべき課題と実社会へのインパクト







インターネットマップ



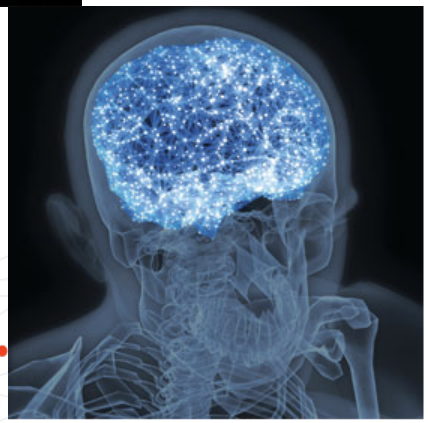
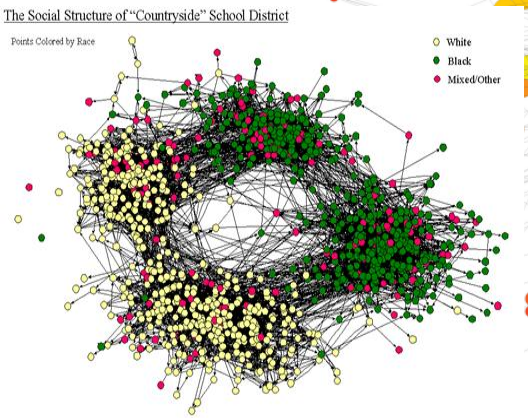


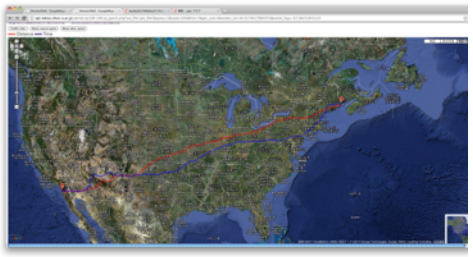
Image: Illustration by Mirko Ilic

ニューラルネットワーク



ソーシャルネットワーク

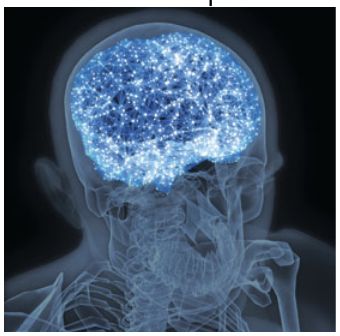
サイバーセキュリティ



交通(時空間)ネットワーク

スモールワールドやスケールフリー性などの特性を持つ超巨大グラフ
(1兆点以上) 上での高速探索技術（アルゴリズム、実装技術、高速計算）

京スパコン: 65536ノード
Graph500: 17977 GTEPS



Human Brain Project

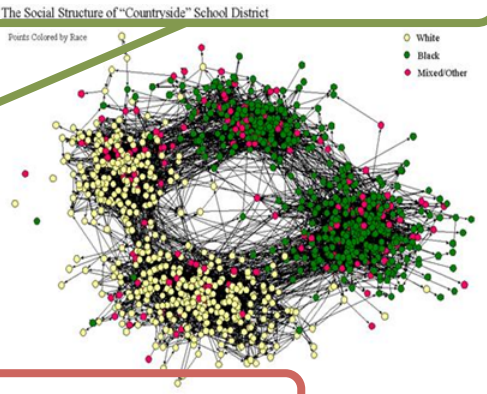
Graph500 (Huge)

Symbolic Network

Graph500 (Large)

Graph500 (Medium)

Twitter (tweets / day)



Graph500 (Small)

Graph500 (Mini)

Graph500 (Toy)

USA-road-d.USA.gr

全米道路ネットワーク

USA-road-d.LKS.gr

USA-road-d.NY.gr

Twitter (バルス:2013/08/02)

1 兆点

1 0億点

1 兆枝

1 0億枝

グラフの枝数

$\log_2(m)$

$\log_2(n)$

グラフの点数

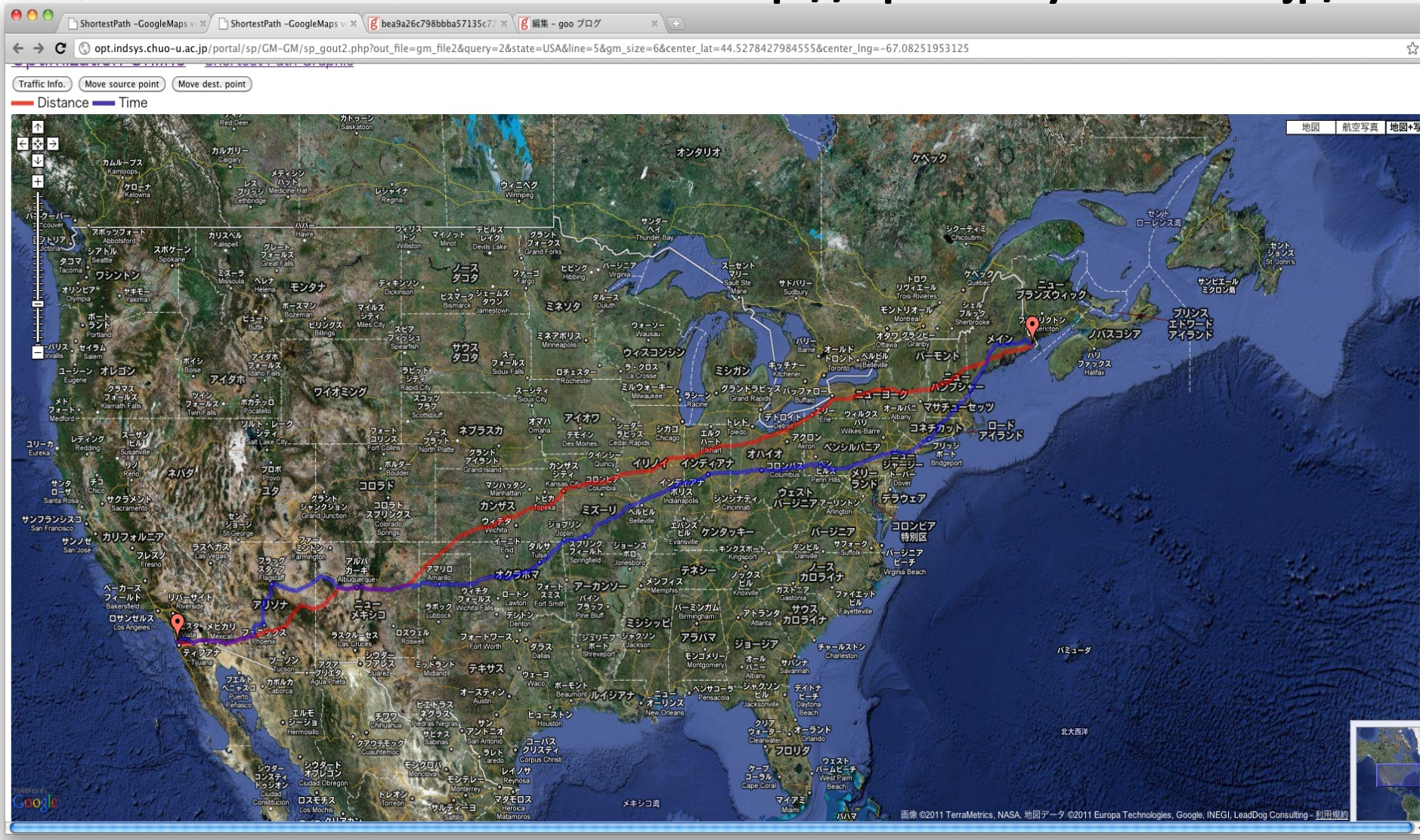
45
40
35
30
25
20
15
20
25
30
35
40
45

Android 携帯 : SONY SO-01F
Snapdragon S4 1.7GHz 4コア: 2GB RAM
1.03GTEPS: 235.06MTEPS/W

全米横断の最短路：2400万点, 5800万枝 → 約2秒で計算(3W)

サンディエゴから北東部のメーン州までの最短路

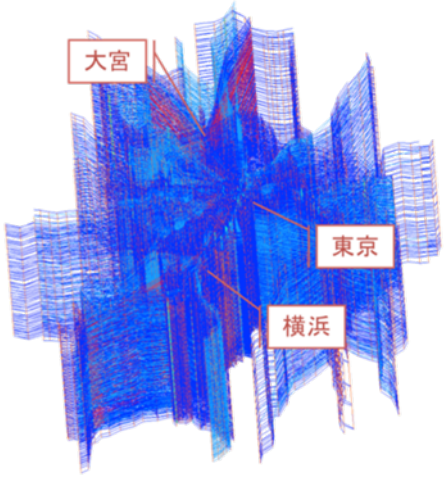
（赤：最短距離、青：最短時間）：http://opt.imi.kyushu-u.ac.jp/



超大規模ネットワークにおける高速探索技術の開発
コアとなるアルゴリズムやソフトウェアのカーネル部分の実装と評価

- 超大規模ネットワークデータ（点数１０億以上)が対象
- 最短路(1対全), グラフ探索(深さ優先、幅優先等)
- グラフ探索の局所的な評価では優先キュー(ヒープ木)を用いる⇒実行時間、メモリ消費量が安定的
- 実行時間(1対全最短路計算: CPU : Intel Xeon 5670 2.93GHz)
  - 全米グラフ 3秒（点数 n = 23,947,347、枝数 m = 58,333,344)
  - 超大規模グラフ 870秒 (点数 n = 10億, 枝数 m = 20億)
- 多くの応用(交通ネットワーク, e-コミュニティのネットワーク、ロジスティクスネットワーク)

			スレッド数	クエリ数	消費メモリMB	実行時間(秒)	実行時間/クエリ(マイクロ秒)	確認リンク数	usec/#arcs	コスト
始点	横浜	4822491	1	100	564.54	5.564	55640	428086	0.1300	19108
終点	東京	757995	8	100	1815.59	2.139	21390	428086	0.0500	19108
始点	新宿	1085224	1	100	564.54	35.794	357940	4046406	0.0885	34493
終点	千葉	4174499	8	100	1815.59	7.610	76100	4046406	0.0188	34493
始点	川崎	661699	1	100	564.54	17.954	179540	2072075	0.0866	30527
終点	大宮	1576725	8	100	1815.59	3.905	39050	2072075	0.0188	30527
始点	東京	2321666	1	100	564.54	10.190	101900	1026584	0.0993	19002
終点	ディ	261784	8	100	1815.59	3.160	31600	1026584	0.0308	19002
始点	沖縄	377	1	72	564.54	257.421	3575292	44595172	0.0802	17264925
終点	北海道	9065116	8	72	1815.59	44.871	623208	44595172	0.0140	17264925



点の重要性

- グラフの特徴を示す指標
  - 中心性、PageRank …
- 中心性指標が広く利用されている
  - 近年は中心性が盛んに研究されている
- 中心性指標にもいくつか種類がある
  - 各点に接続されている枝数（Degree Centrality）
  - 各点が最短路に含まれる回数（Betweenness Centrality）
  - 最も遠い点までの最短距離（Graph Centrality）
  - 各点までの最短距離（Closeness Centrality）

中心性の定義

- Closeness Centrality

$$C_C(v) = \frac{1}{\sum_{t \in V} d_G(v,t)}$$

v-t 間最短路長

- Graph Centrality

$$C_G(v) = \frac{1}{\max_{t \in V} d_G(v,t)}$$

- Stress Centrality

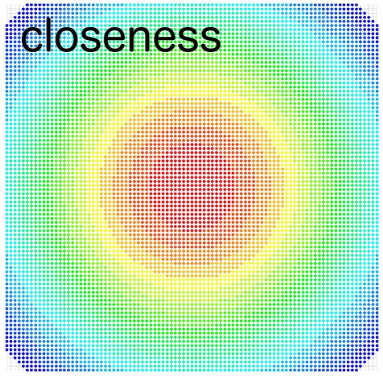
$$C_S(v) = \sum_{s \neq v \neq t \in V} \sigma_{st}(v)$$

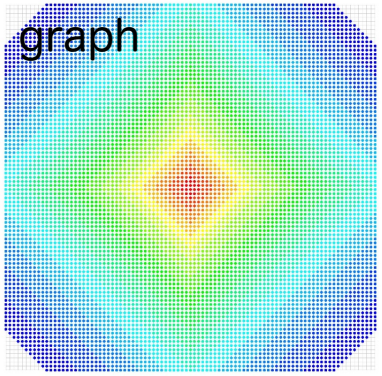
v を通る s-t 間最短路数

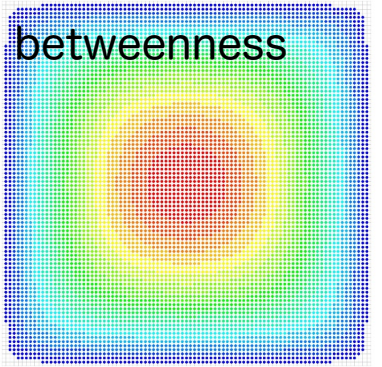
- Betweenness Centrality

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

s-t 間最短路数



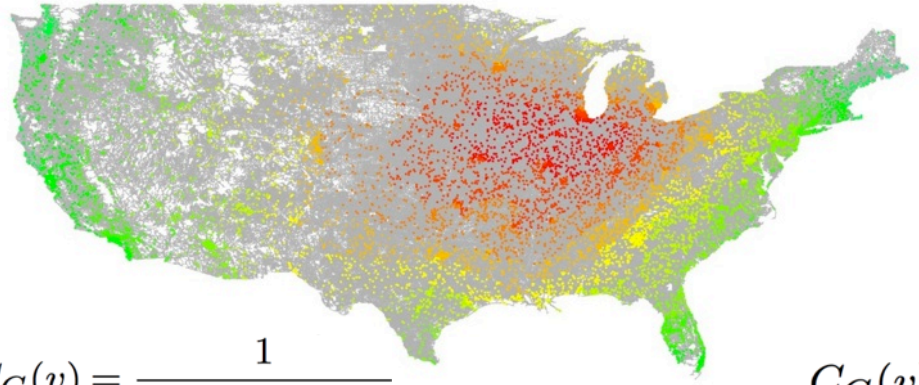




全米グラフに対する重要性(中心性)の計算

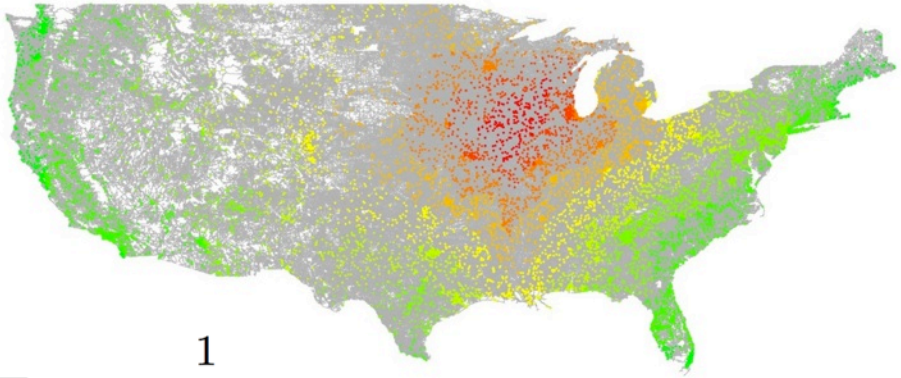
- ネットワーク内での各点の**重要度**を推定
- **各点の周辺、及び広域内における影響**(情報の伝播力)を推定する
- 超大規模ネットワークデータ上での短時間に大量のグラフ探索を行う必要がある

Closeness



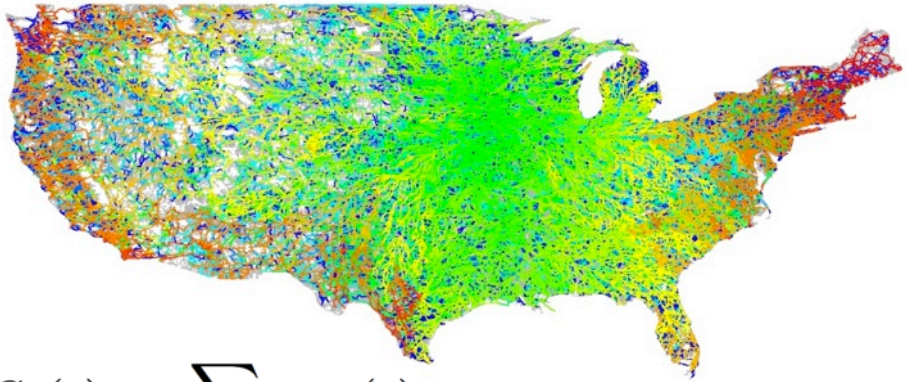
$$C_C(v) = \frac{1}{\sum_{t \in V} d_G(v,t)}$$

Graph



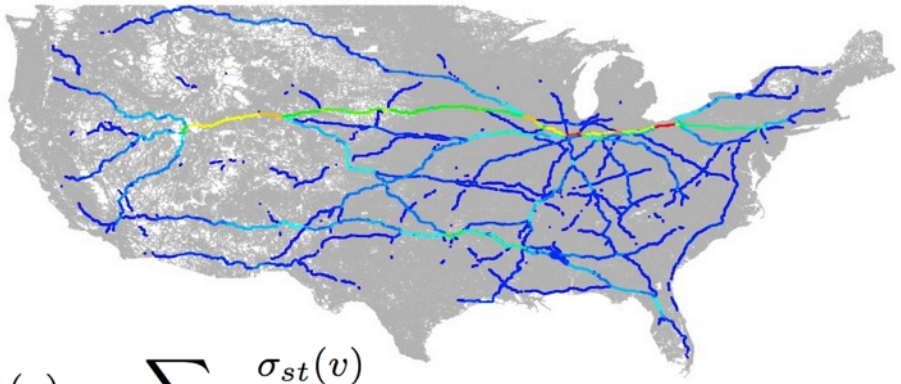
$$C_G(v) = \frac{1}{\max_{t \in V} d_G(v,t)}$$

Stress(対数)



$$C_S(v) = \sum_{s \neq v \neq t \in V} \sigma_{st}(v)$$

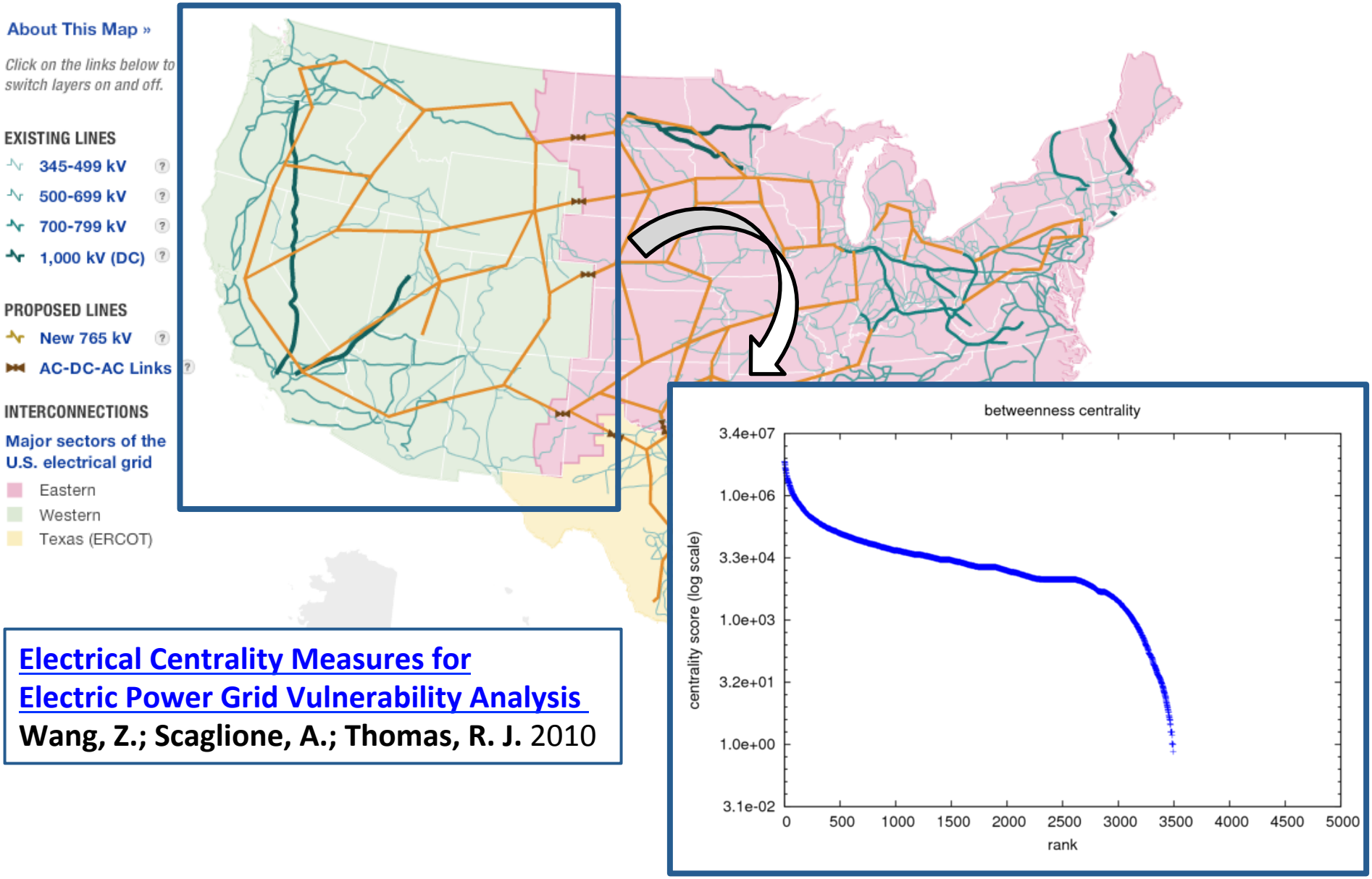
Betweenness



$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

The U.S. Electric Grid

http://www.npr.org/templates/story/story.php?storyId=110997398



Electrical Centrality Measures for Electric Power Grid Vulnerability Analysis
Wang, Z.; Scaglione, A.; Thomas, R. J. 2010

全米道路ネットワークに対する全対全最短路問題

点数 n = 23,947,347, 枝数 m = 58,333,344

4-way Opteron 6174 2.3 GHz (12 cores x 4), 256 GB, GCC-4.6.0

アルゴリズム	全対全最短路	高速化率
Dijkstra's algorithm	512 年	--
Δ-stepping	9.1 年	60 倍
Multi-Level Buckets	4.9 年	110 倍
MSLC + NUMAを考慮した最適化	7.8 日	24,000 倍

各2点間の最短路長を
1.1ns で求めることができる

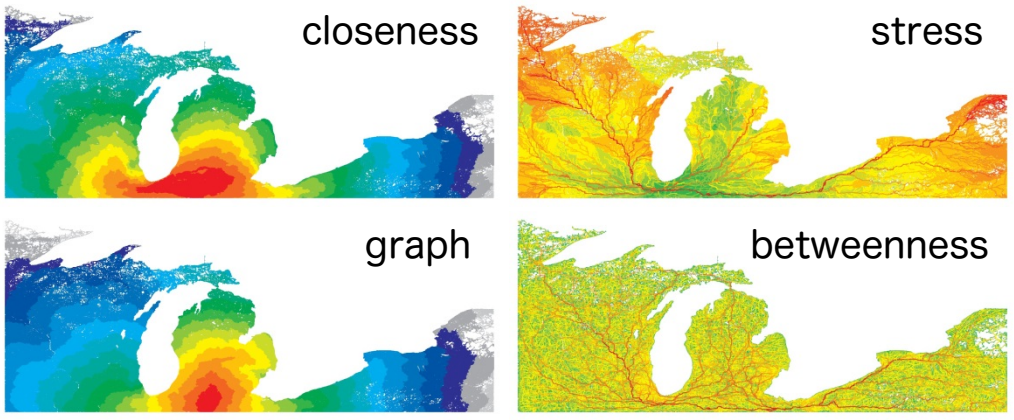
厳密な
全対全最短路長を
約7.8日で計算

五大湖周辺道路ネットワーク最短路を用いた中心性計算

点数 n = 2,758,119, 枝数 m = 6,885,658

4-way Opteron 6174 2.3 GHz (12 cores x 4), 256 GB, GCC-4.6.0

重みを考慮しない中心性：19.35 時間



既存の GraphCT では
BC のみの計算で
20.6 日かかる

NYデータに対する中心性(BC)の計算結果(最短路問題)

○サーバ1 (2CPU x 4 コア = 8 コア)
実行時間 1957.652 秒 : 8スレッド

CPU : AMD Opteron 2356 2.3GHz x 2
メモリ : 32GB
OS : Vine Linux 5.1 for X86_64

○サーバ2 (1CPU x 4 コア = 4 コア)
実行時間 2058.443 秒 : 4スレッド

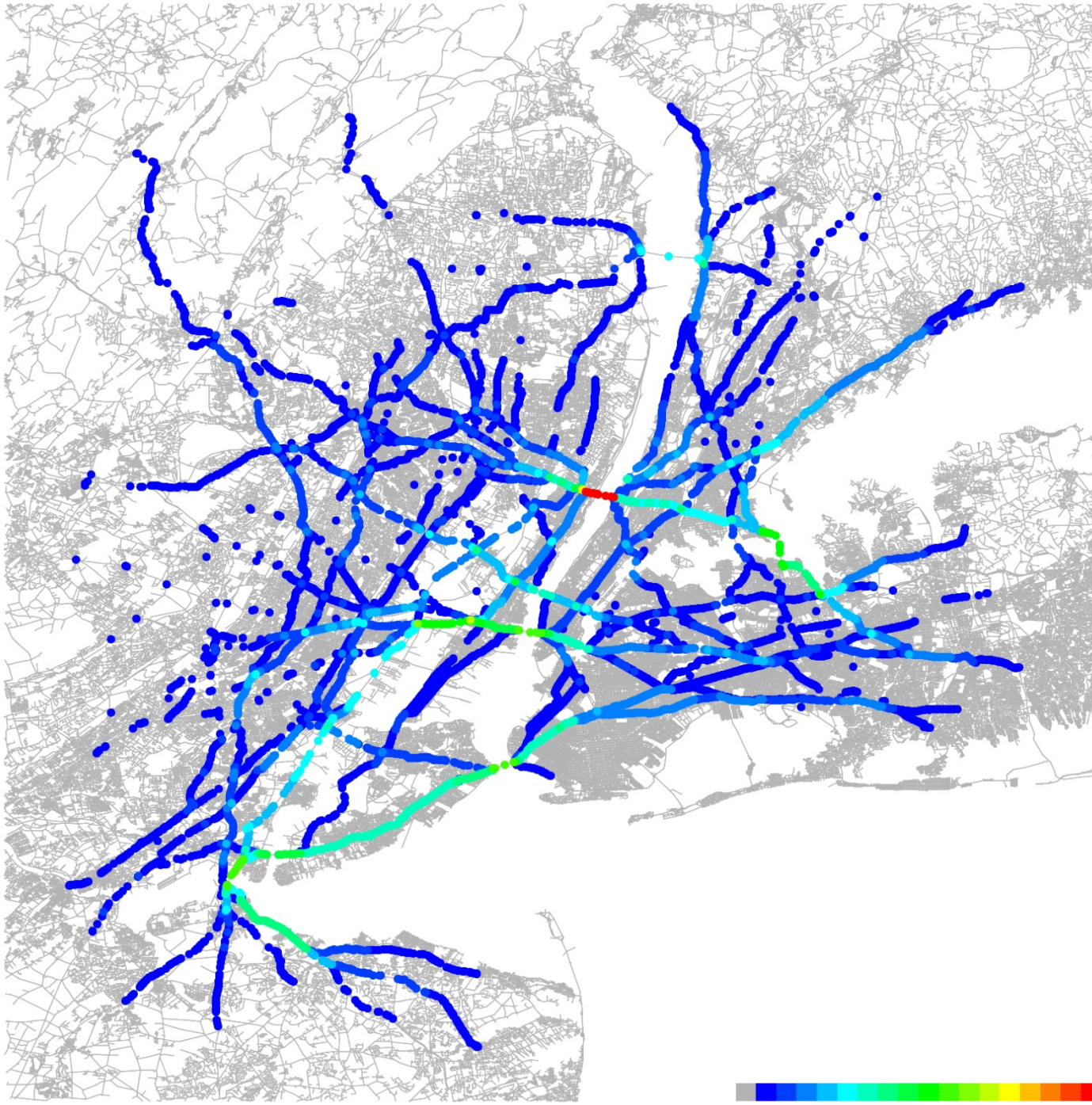
CPU : Intel Core i7 965 3.2GHz x 1
メモリ : 12GB
OS : CentOS 5.5 for x86_64

○サーバ3 (4 CPU x 6 コア = 24 コア)
実行時間 632.859 秒 : 24スレッド

CPU : AMD Opteron 8439 2.80GHz x 4
Memory : 128GB
OS : Fedora 13 for x86_64

○ サーバ4 (2 CPU x 4 コア = 8 コア)
実行時間 1319.054 秒 : 8スレッド

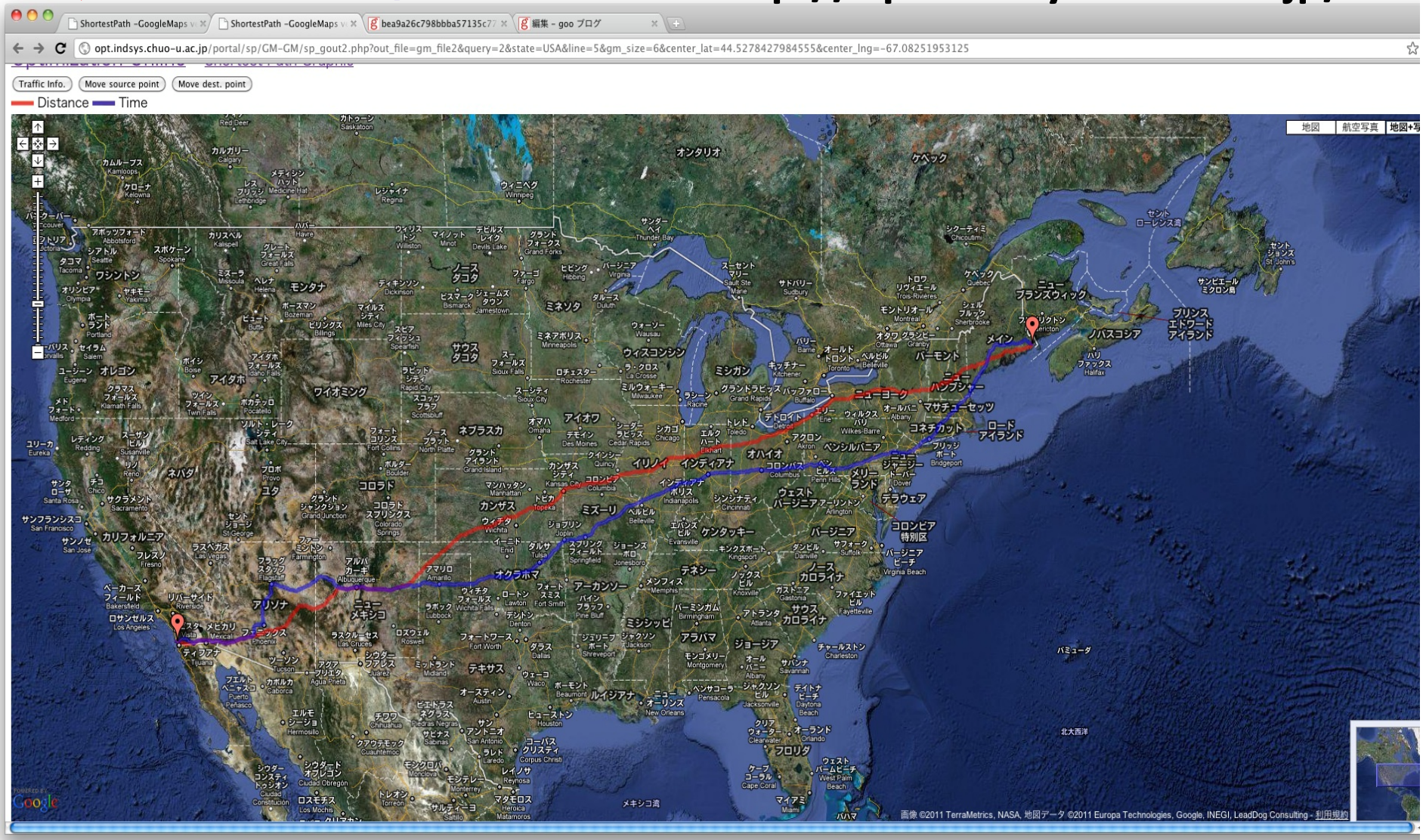
CPU : Intel Xeon 5550 2.66GHz x 2
Memory : 72GB
OS : Fedora 13 for x86_64



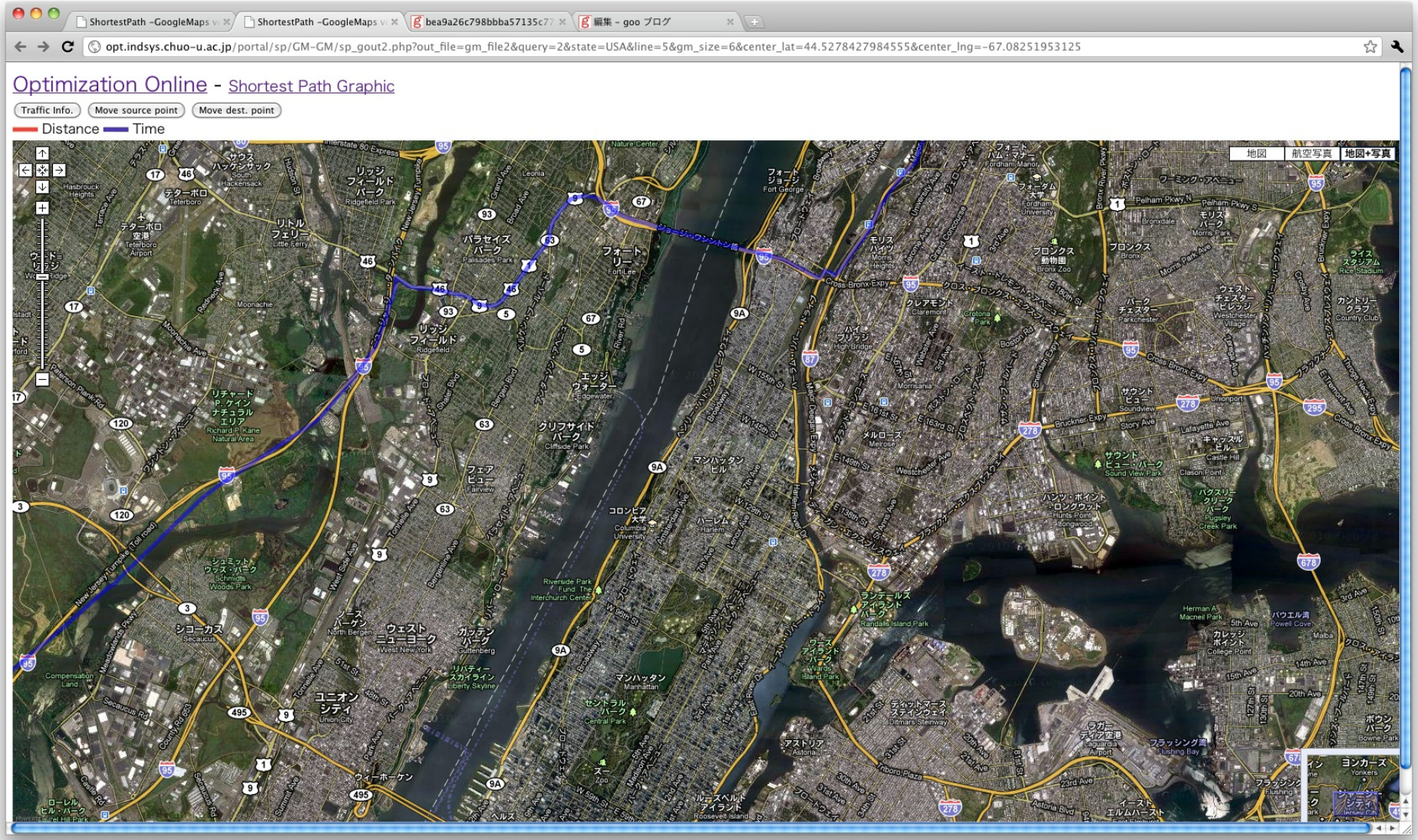
全米横断の最短路：2400万点, 5800万枝 → 約2秒で計算(3W)

サンディエゴから北東部のメーン州までの最短路

（赤：最短距離、青：最短時間）：http://opt.imi.kyushu-u.ac.jp/



ジョージ・ワシントン橋, ニューヨークを通過する全米横断最短路



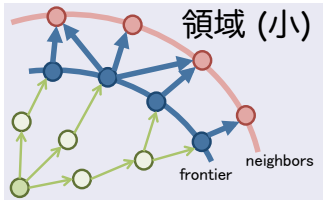
Graph500・Green Graph500

○ Graph500・Green Graph500 ベンチマーク

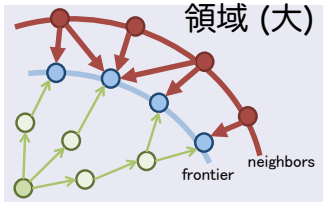
- パラメータ SCALE と edgefactor (=16) から、点数 $n=2^{SCALE}$, 枝数 m=edgefactor・n となる Kroncekr graph を生成
- 幅優先探索 (BFS) での1秒辺りの探索枝数 TEPS により、Graph500 リストを作成
- 消費電力あたりの TEPS（TEPS/W) により、Green Graph500 リストを作成

○ 高速な幅優先探索の実装 BFS

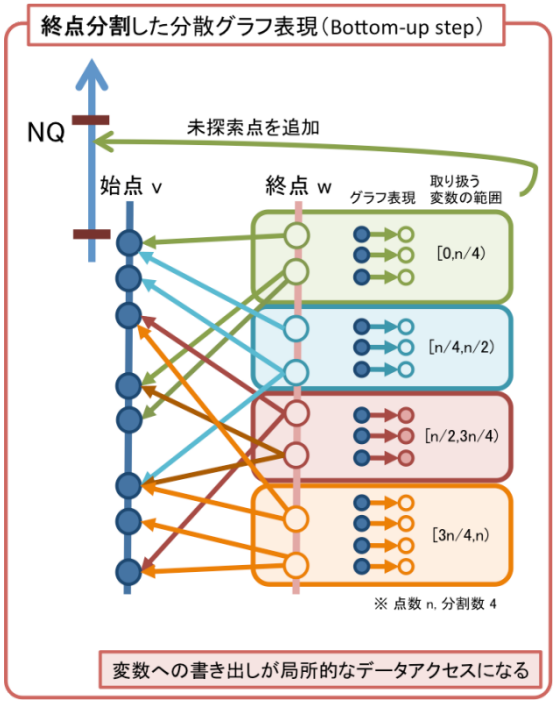
- 計算機性能を引き出す汎用的な高速化



Top-down 探索



Bottom-up 探索



○ Green Graph500 (June 2013)

Rank	MTEPS/W	Site	Machine	G500 rank	Scale	GTEPS	Nodes
1	64.12	Chuo University	GraphCREST-Tegra	132	20	0.153885	1
2	53.82	Chuo University	GraphCREST-Intel-NUC	110	23	1.08175	1
3	53.47	Chuo University	GraphCREST-Mac-mini	90	24	1.94104	1
4	52.02	Chuo University	GraphCREST-MBA13	105	23	1.22761	1
5	51.62	Chuo University	GraphCREST-Retina15	89	24	1.98735	1
6	39.29	Changsha, China	TH-IVB-FEP/C	-	26	9.74402	1
7	32.25	Chuo University	GraphCREST-NEXUS10	133	20	0.118688	1
8	30.41	Tokyo Institute of Technology	ULP2012 ultra-green oil-cooled server	76	24	5.12372	1
9	20.58	Chuo University	GraphCREST-Blade	72	25	6.14248	1
10	18.68	Chuo University	GraphCREST-SBep2.9	67	25	6.77685	1
11	17.39	Chuo University	GraphCREST-4way5Bep2.4	57	26	11.1148	1
12	10.95	Chuo University	GraphCREST-Wex40	58	26	11.061	1
13	2.43	The University of British Columbia	Alkindi	106	28	1.20576	1
14	1.89	University of Luxembourg	Viridis HPC@Uni.lu	-	21	0.389643	32
15	0.20	Swiss National Supercomputing Center	Todi	49	29	15.5983	272

上位独占

Graph500ベンチマークの実行順

Kernel 1

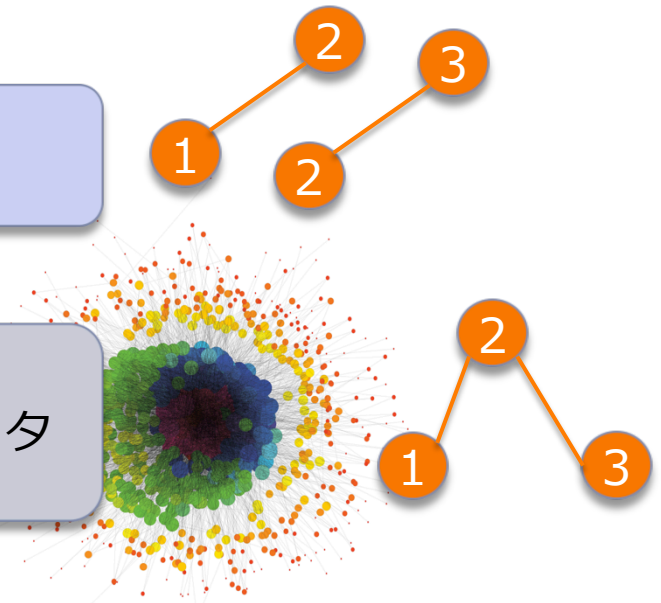
Kernel 2

Graph Generation
（枝リストを生成）

Graph Construction
(枝リストをBFSで探索するためのデータ(CSR or CSC)に変換)

Breadth First Search

Validation
（５つのルールでBFSツリーを検証）



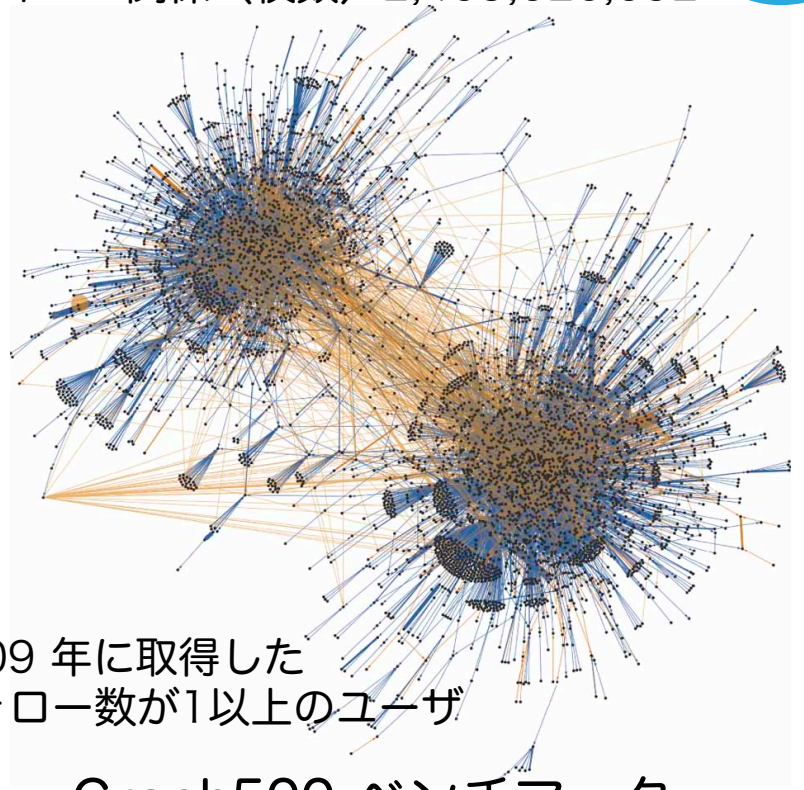
始点をランダムに64個選択

64回
繰り返す

※CSR: Compressed Sparse Row, CSC: Compressed Sparse Column

Graph500技術の応用
Twitter ネットワークの解析

- フォロー・ネットワーク
  - ユーザ数　（点数）　41,652,230
  - フォロー関係（枝数）2,405,026,092



2009 年に取得した
フォロー数が1以上のユーザ

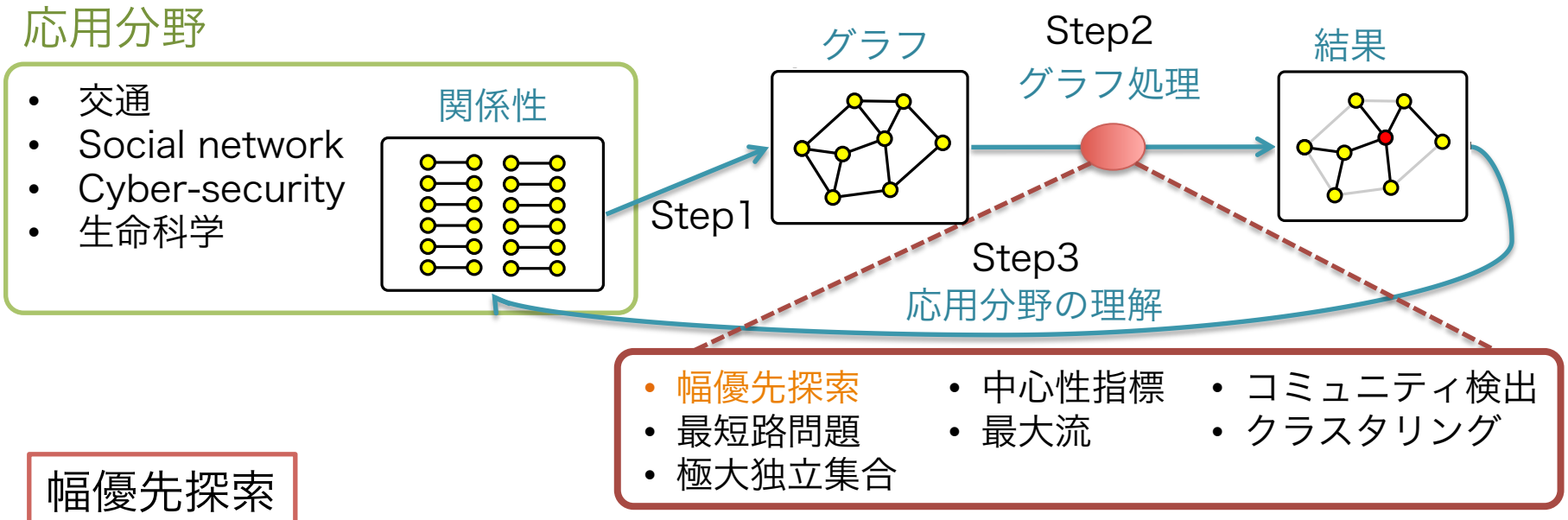
- Graph500 ベンチマーク
  - 幅優先探索の性能「1 秒間に通過した枝数 TEPS」を用いて、コンピュータの性能を比較する

ユーザ 21,804,357 からの幅優先探索の結果

ホップ数	ユーザ数	割合 (%)	累積割合 (%)
0	1	0.00	0.00
1	7	0.00	0.00
2	6,188	0.01	0.01
3	510,515	1.23	1.24
4	29,526,508	70.89	72.13
5	11,314,238	27.16	99.29
6	282,456	0.68	99.97
7	11536	0.03	100.00
8	673	0.00	100.00
9	68	0.00	100.00
10	19	0.00	100.00
11	10	0.00	100.00
12	5	0.00	100.00
13	2	0.00	100.00
14	2	0.00	100.00
15	2	0.00	100.00
合計	41,652,230	100.00	-

0.069 秒で探索可能
⇒ 21.28 GTEPS（10^9 TEPS）

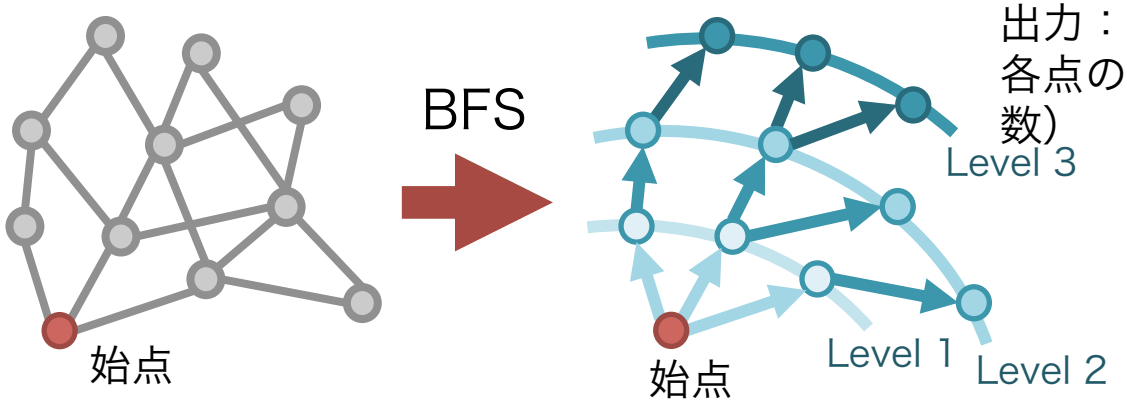
グラフ処理と幅優先探索



幅優先探索

- 最も重要かつ基本的なグラフ処理の一つ
- 他のアルゴリズムのサブルーチン
- 演算量に比べて、不連続なメモリアクセスが多いことが問題

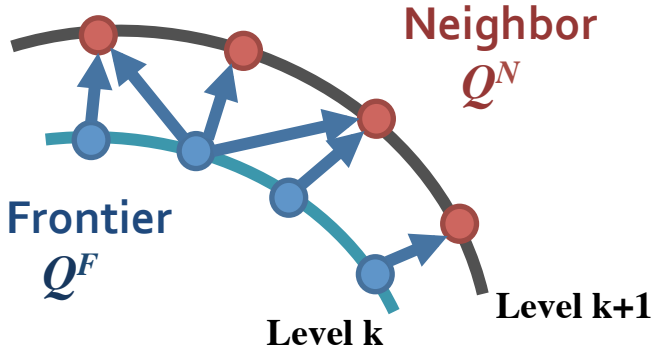
入力：
グラフと始点



Level-synchronized parallel BFS (**Top-down**)

- Started from source vertex and executes following two phases for each level

Traversal ⋯ finds **unvisited adjacency vertices** from current frontier Q^F and append to neighbor Q^N



Swap ⋯ swaps the frontier Q^F and the neighbor Q^N for next level

Algorithm 1: Level-synchronized Parallel BFS.

Input : $G = (V, A)$: unweighted directed graph.
s : source vertex.
Variables: Q^F : *frontier* queue.
Q^N : *neighbor* queue.
visited : vertices already visited.
Output : $\pi(v)$: predecessor map of BFS tree.

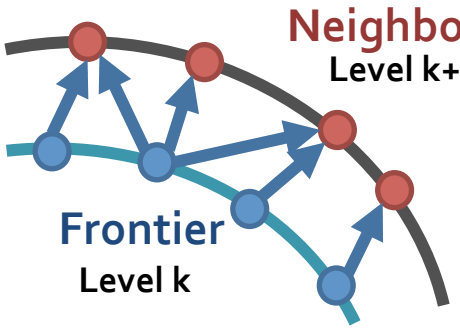
```
1  π(v) ← −1, ∀v ∈ V
2  π(s) ← s
3  visited ← {s}
4  Q^F ← {s}
5  Q^N ← ∅
6  while Q^F ≠ ∅ do
7      for v ∈ Q^F in parallel do                          ┐
8          for w ∈ A(v) do                                 │ Traversal
9              if w ∉ visited atomic then                  │
10                 π(w) ← v                                │
11                 visited ← visited ∪ {w}                 │
12                 Q^N ← Q^N ∪ {w}                         ┘
13     Q^F ← Q^N                                           ┐ Swap
14     Q^N ← ∅                                             ┘
```

Hybrid-BFS (Direction-optimizing BFS)

Chooses one from **Top-down** or **Bottom-up** for frontier size at each level

Top-down algorithm

- Efficient for **small**-frontier
- Uses **out-going** edges



Input : Directed graph $G = (V, A^{\mathcal{F}})$, Queue Q^F
Data : Queue Q^N, visited, Tree $\pi(v)$

$Q^N \leftarrow \emptyset$
for $v \in Q^F$ **in parallel do**
  for $w \in A^{\mathcal{F}}(v)$ **do**
    if $w \notin$ *visited* **atomic then**
      $\pi(w) \leftarrow v$
      visited $\leftarrow$ *visited* $\cup \{w\}$
      $Q^N \leftarrow Q^N \cup \{w\}$
$Q^F \leftarrow Q^N$

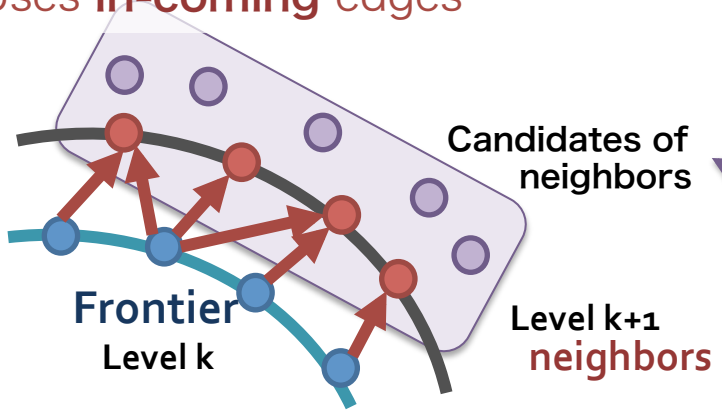
Current frontier

Unvisited neighbors

Beamer2012

Bottom-up algorithm

- Efficient for **large**-frontier
- Uses **in-coming** edges



Input : Directed graph $G = (V, A^{\mathcal{B}})$, Queue Q^F
Data : Queue Q^N, visited, Tree $\pi(v)$

$Q^N \leftarrow \emptyset$
for $w \in V \setminus$ *visited* **in parallel do**
  for $v \in A^{\mathcal{B}}(w)$ **do**
    if $v \in Q^F$ **then**
      $\pi(w) \leftarrow v$
      visited $\leftarrow$ *visited* $\cup \{w\}$
      $Q^N \leftarrow Q^N \cup \{w\}$
      break
$Q^F \leftarrow Q^N$

Candidates of neighbors

Current frontier

Skips unnecessary edge traversal

Hybrid-BFS (Direction-optimizing BFS)

Chooses one from **Top-down** or **Bottom-up**
for a number of traversed edges at each level

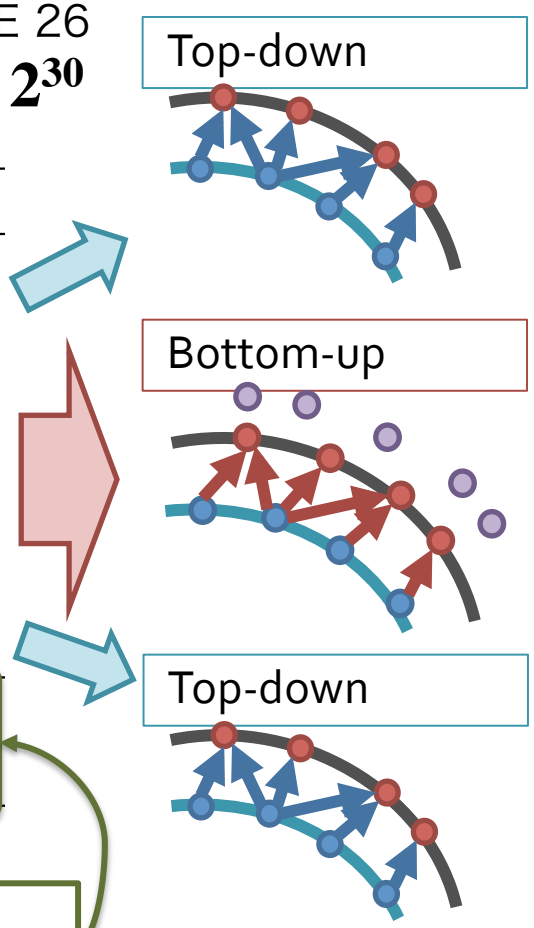
Beamer2012

Number of traversal edges of Kronecker graph with SCALE 26

$|V| = 2^{26}, |E| = 2^{30}$

Distance from source

Level	Top-down	Bottom-up	Hybrid
0	2	2,103,840,895	2
1	66,206	1,766,587,029	66,206
2	346,918,235	52,677,691	52,677,691
3	1,727,195,615	12,820,854	12,820,854
4	29,557,400	103,184	103,184
5	82,357	21,467	21,467
6	221	21,240	227
Total	2,103,820,036	3,936,072,360	65,689,631
Ratio	100.00%	187.09%	3.12%
	$= \|E\|$		



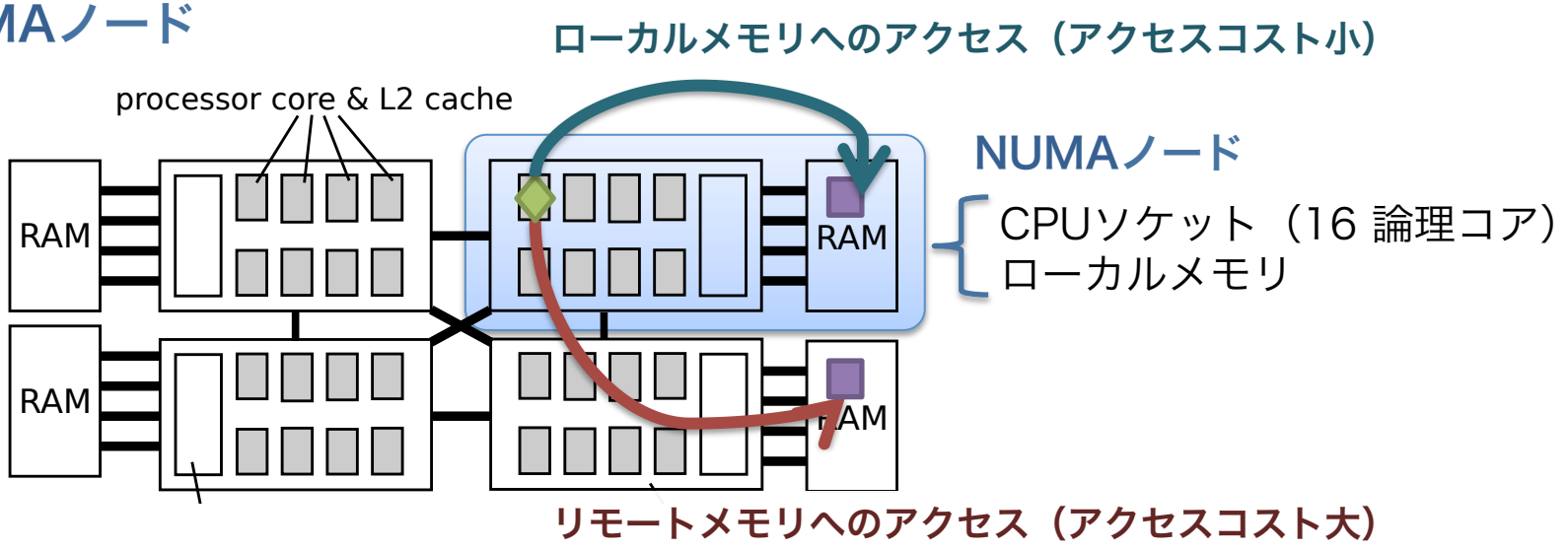
Hybrid-BFS reduces
unnecessary edge traversals

NUMA アーキテクチャの構成

- 4-way Intel Xeon E5-4640 (Sandybridge-EP)
  - 4 (CPU ソケット数)
  - 8 (CPU ソケットあたりの物理コア数)
  - 2 (物理コアあたりのスレッド数)

最大
4 x 8 x 2 = 64 スレッド
（ハイパースレッディング）

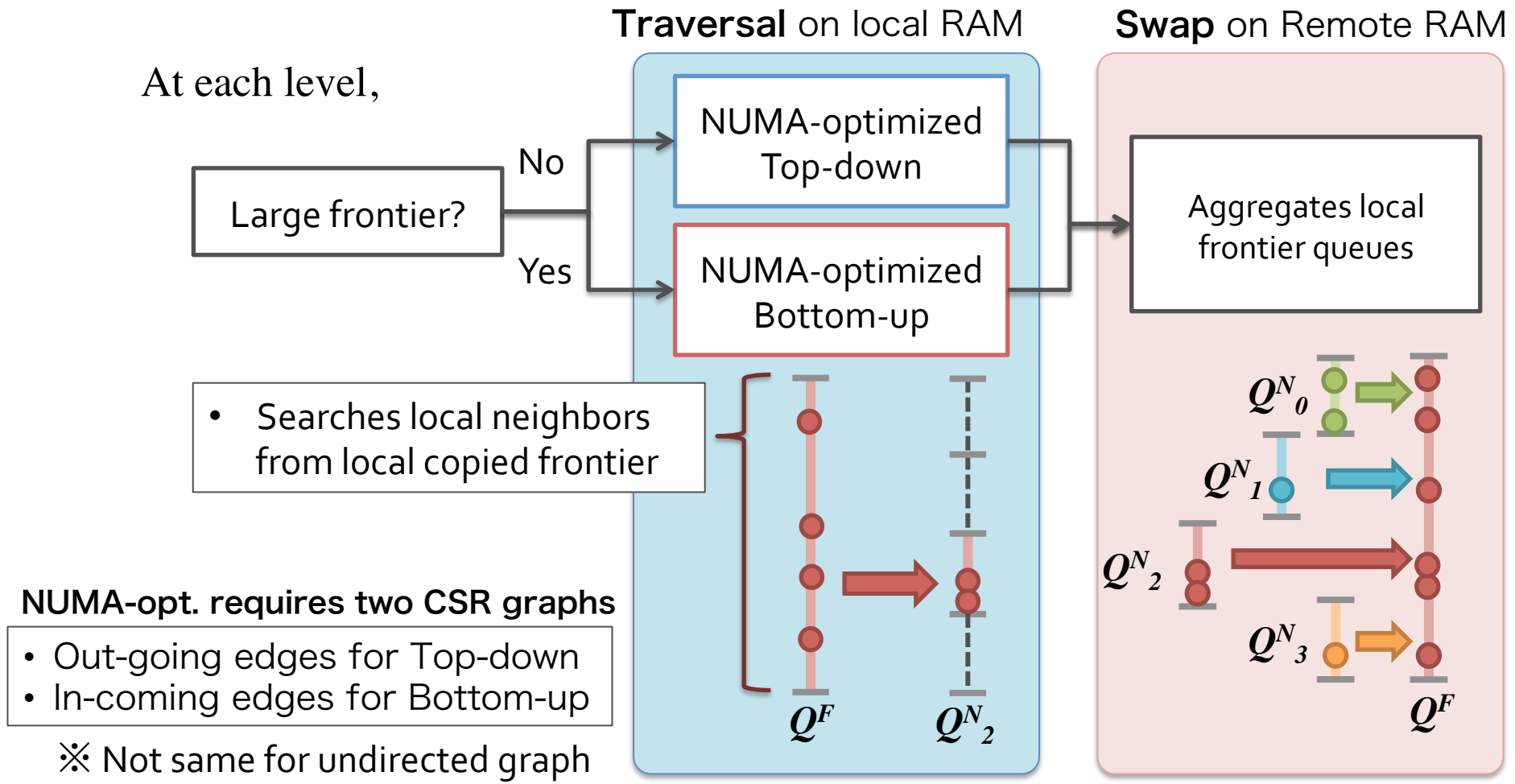
NUMAノード



データアクセスが不均一であるため
- 並列時の偏りが生じるため、並列効率を向上することが難しい
- 本来の性能を予想することが難しい

NUMA-optimized BFS

- Clearly separated to accessing for local and remote memory
  - Edge traversal on Local RAM
  - All-gather of local queues and bitmaps for Remote RAM



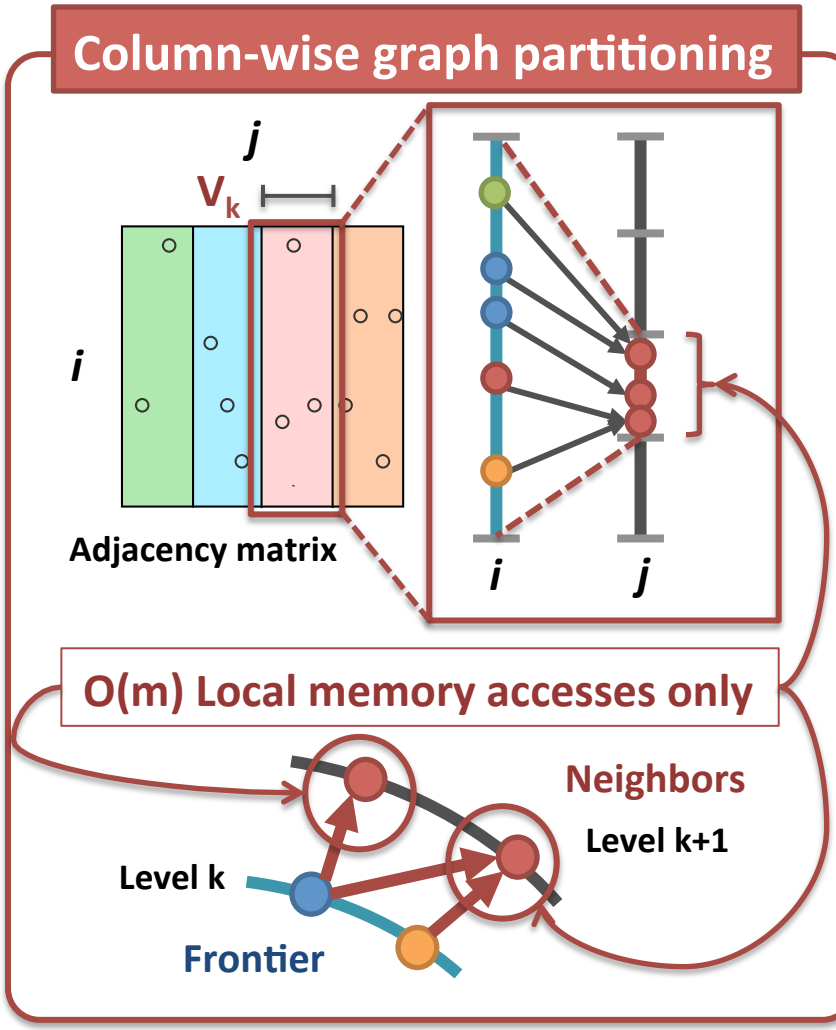
NUMA-opt. requires two CSR graphs

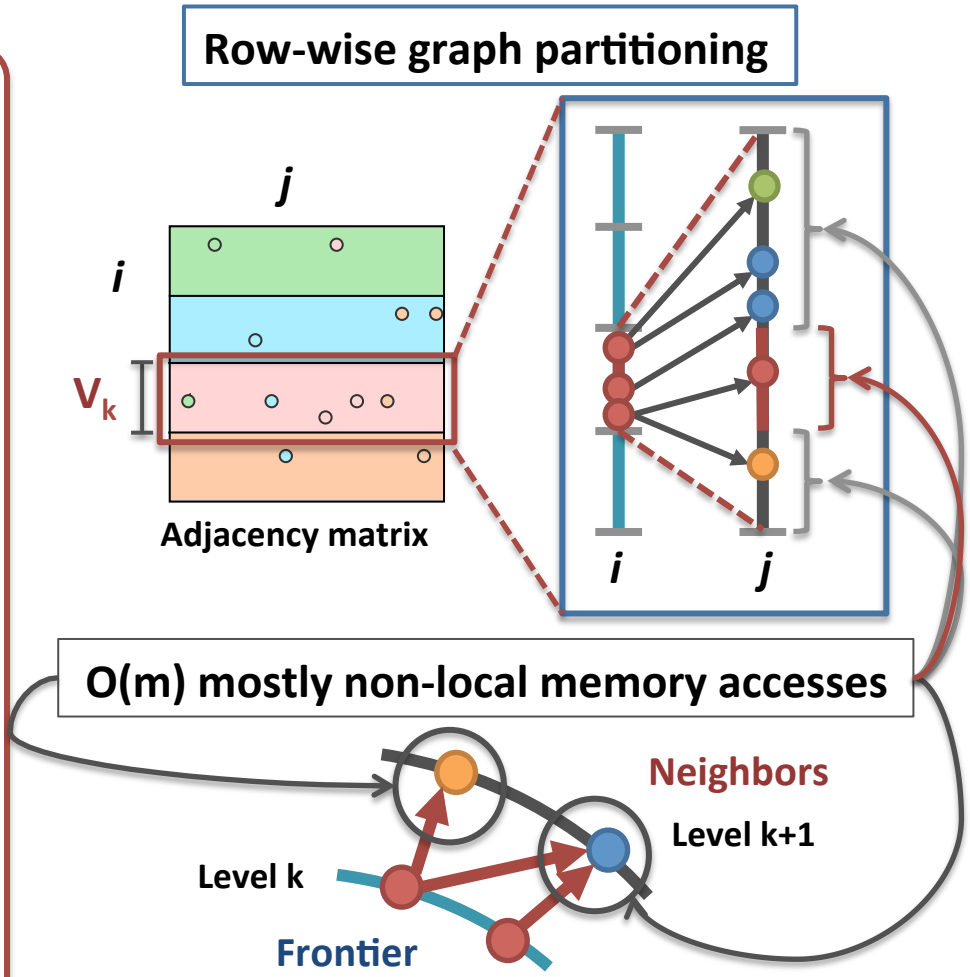
- Out-going edges for Top-down
- In-coming edges for Bottom-up

※ Not same for undirected graph

NUMA-opt. Column-wise Graph Partitioning

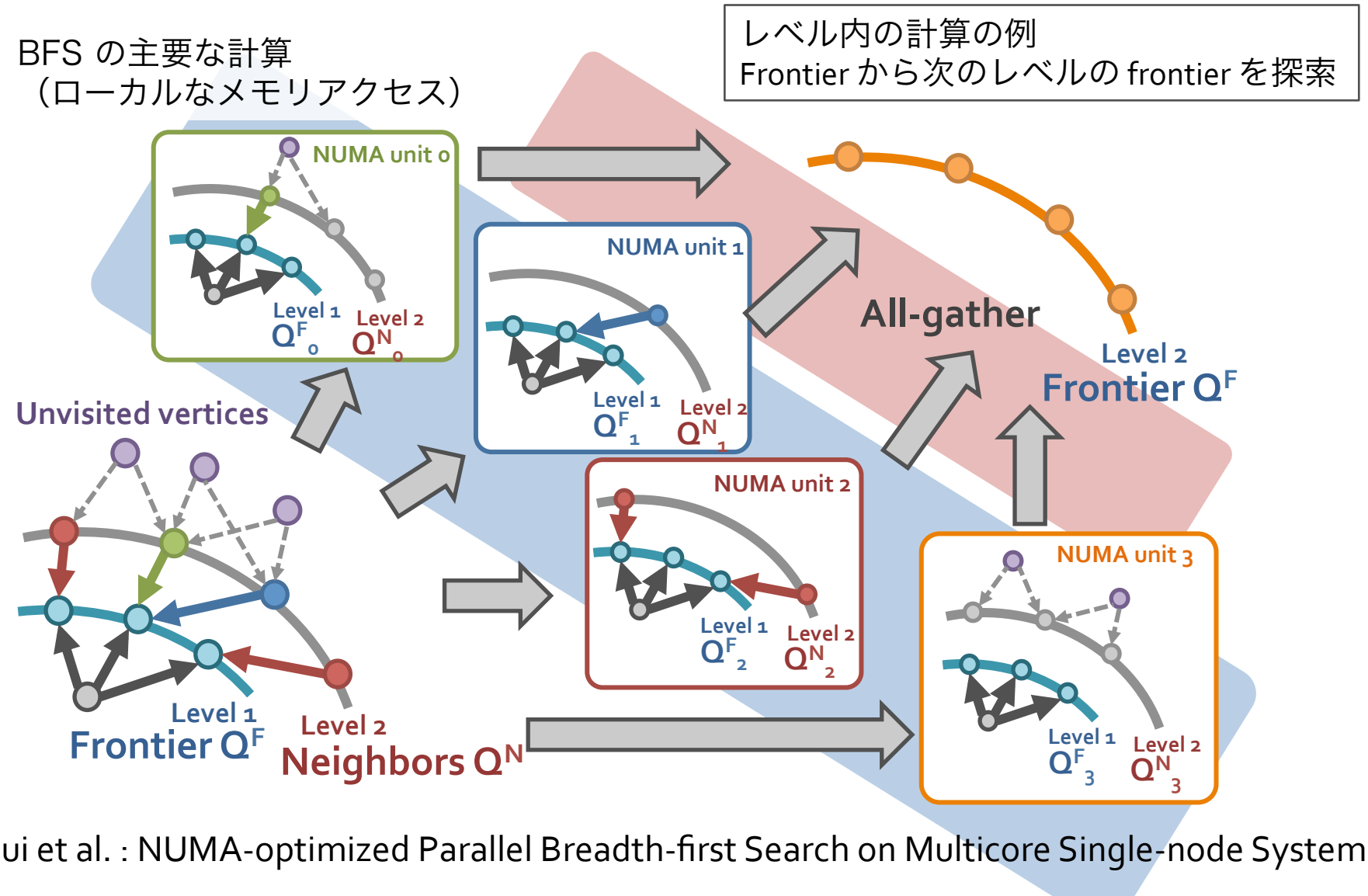
- divides G=(V,A) into partial $G_k=(V_k, A_k)$ and binds local RAM k.
  - A_k is a set of adjacency list that holds incoming edges to V_k.





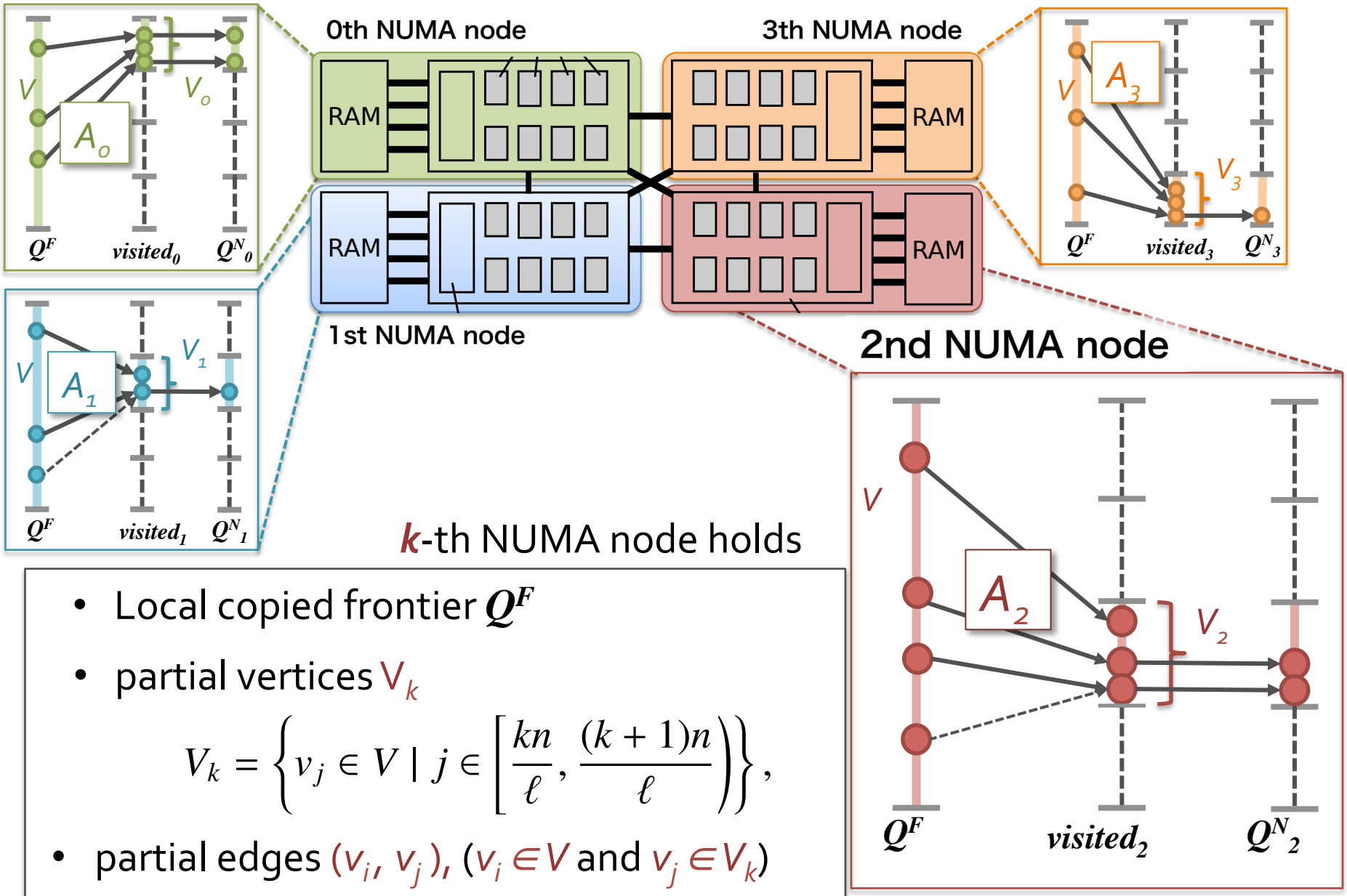
NUMA-opt. parallel hybrid BFS (Graph500 1ノード最速)

- 計算の大部分をローカルなメモリアクセス上で実行



Y. Yasui et al. : NUMA-optimized Parallel Breadth-first Search on Multicore Single-node System

Local Edge Traversal



***k*-th NUMA node holds**

- Local copied frontier $\boldsymbol{Q^F}$
- partial vertices V_k

$$V_k = \left\{ v_j \in V \mid j \in \left[\frac{kn}{\ell}, \frac{(k+1)n}{\ell} \right) \right\},$$

- partial edges (v_i, v_j), ($v_i \in V$ and $v_j \in V_k$)

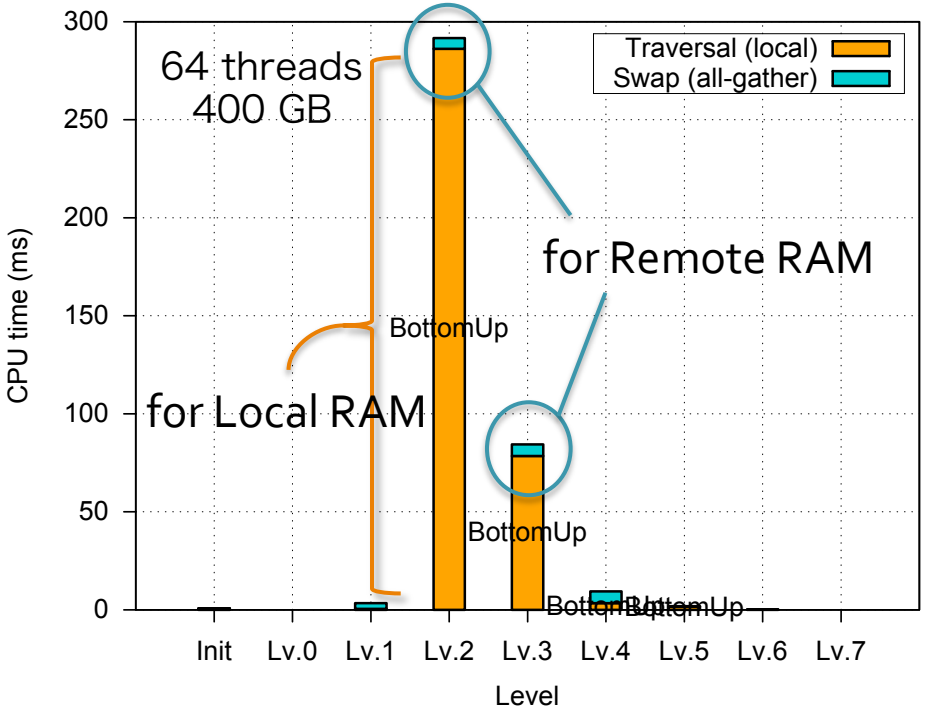
Intel Xeon v.s. SGI Altix UV1000

- **Graph500 (SCALE29)**
  - 536.9 million vertices, 8.59 billion edges

Median TEPS: 21.81 GE/s

Intel Xeon (Sandybridge-EP arch.)

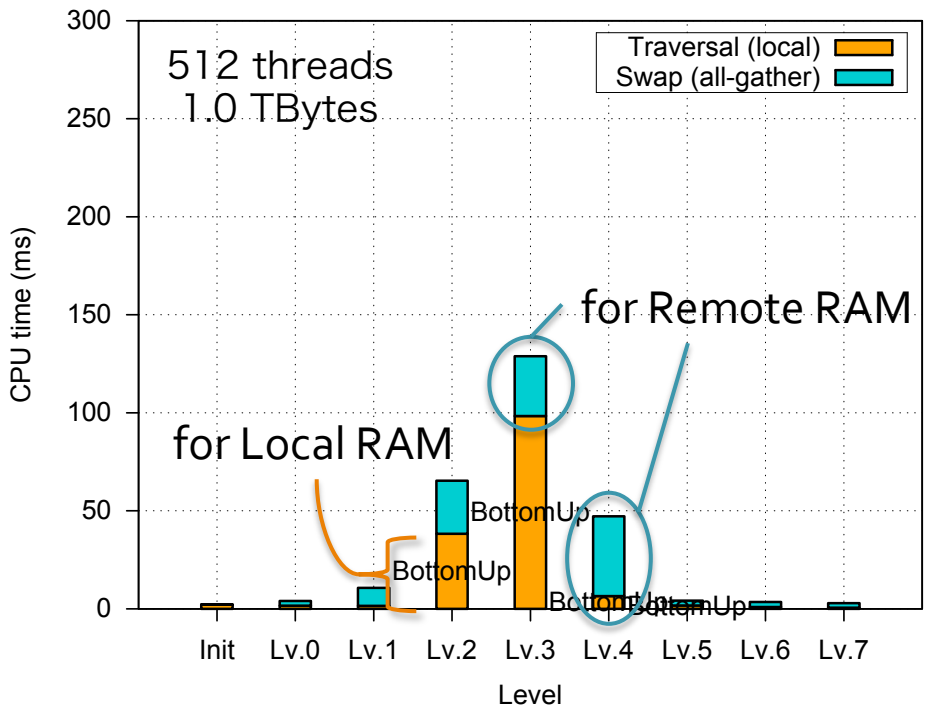
Intel Xeon E5-4650 @ 2.70GHz
64-threads = 8-cores x 2-threads x 4-nodes
512 GBytes RAM



Median TEPS: 31.81 GE/s

SGI Altix UV1000 (Westmere-EX arch.)

Intel Xeon E7-8837 @ 2.67GHz
512-threads = 8-cores x 64-nodes
4.0 TBytes RAM



Intel Xeon v.s. SGI Altix UV1000

- **Graph500 (SCALE30)**
  - 1.07 billion vertices, 17.18 billion edges

Rank.50
Fastest of single-node
on current list

Median TEPS: 37.70 GE/s

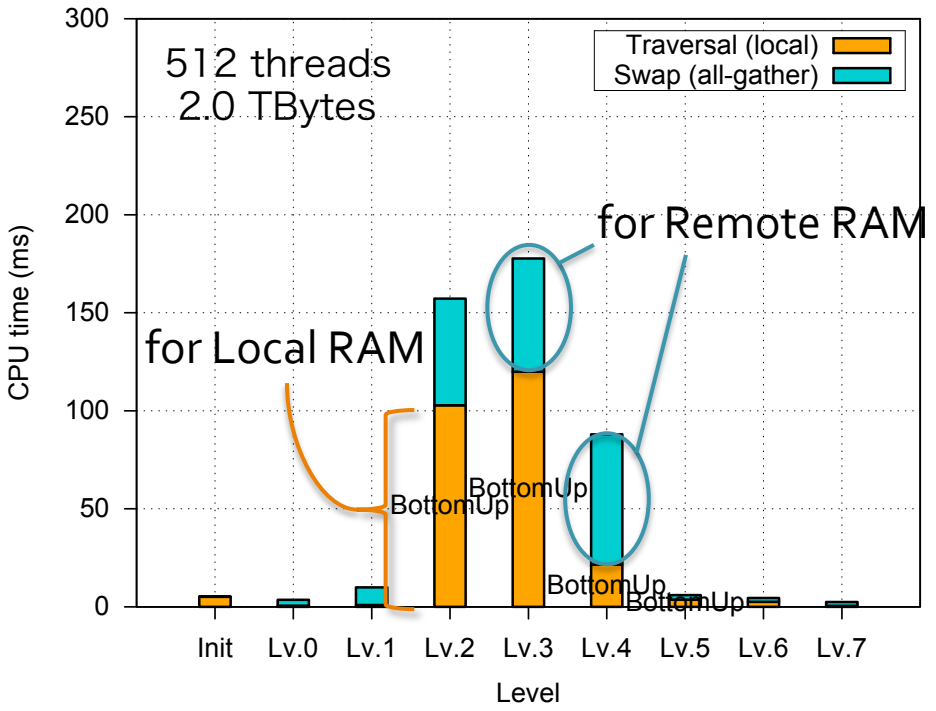
Intel Xeon (Sandybridge-EP arch.)

Intel Xeon E5-4650 @ 2.70GHz
64-threads = 8-cores x 2-threads x 4-nodes
512 GBytes RAM

Out of memory

SGI Altix UV1000 (Westmere-EX arch.)

Intel Xeon E7-8837 @ 2.67GHz
512-threads = 8-cores x 64-nodes
4.0 TBytes RAM



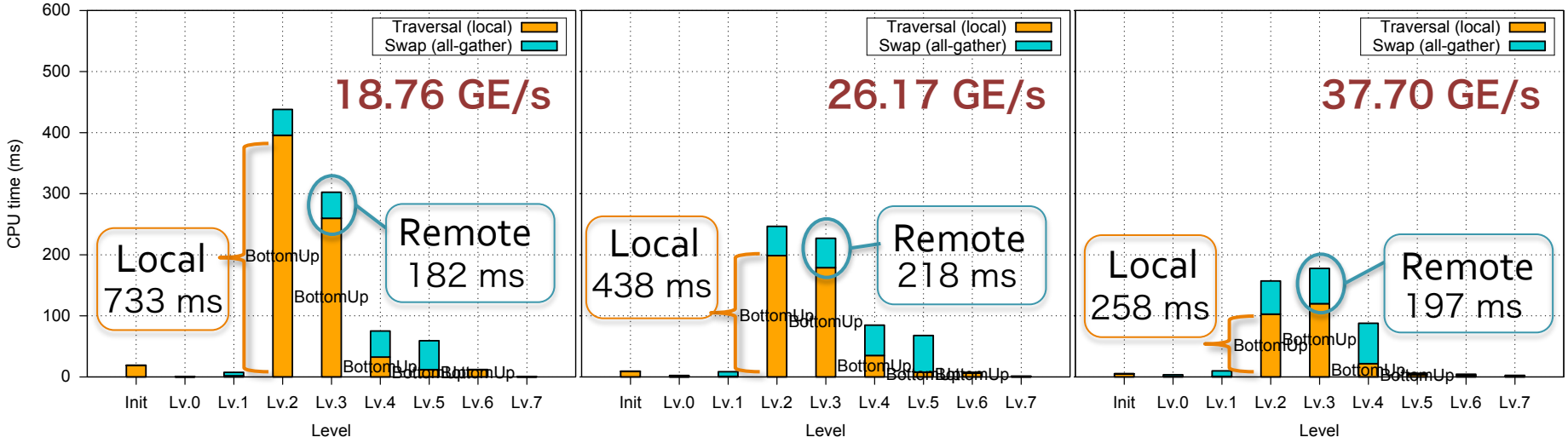
Strong scaling on SGI Altix UV1000

- Graph500 (SCALE30)
  - 1.07 billion vertices, 17.18 billion edges

Rank.50
Fastest of single-node
on current list

- As the number of threads increases,
  - local traversal on Local memory : **Improvement**
  - swap operation on Remote memory : **Keep**

L : R = 80% : 20%　　L : R = 67% : 33%　　L : R = 57% : 43%

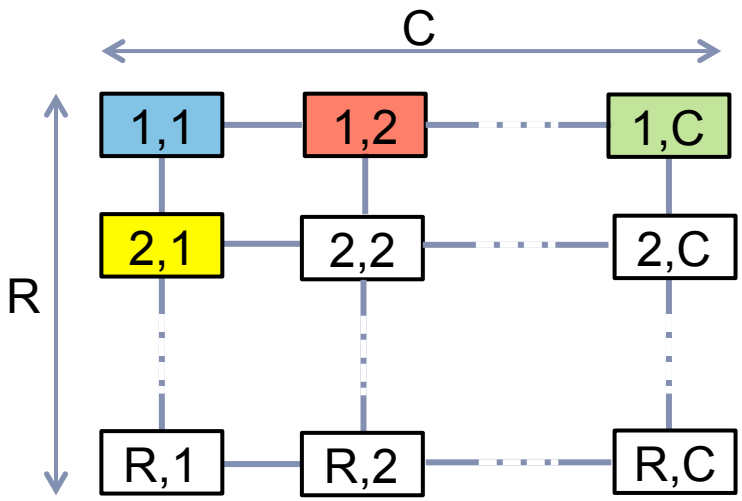


128 threads　　**256 threads**　　**512 threads (one-rack)**

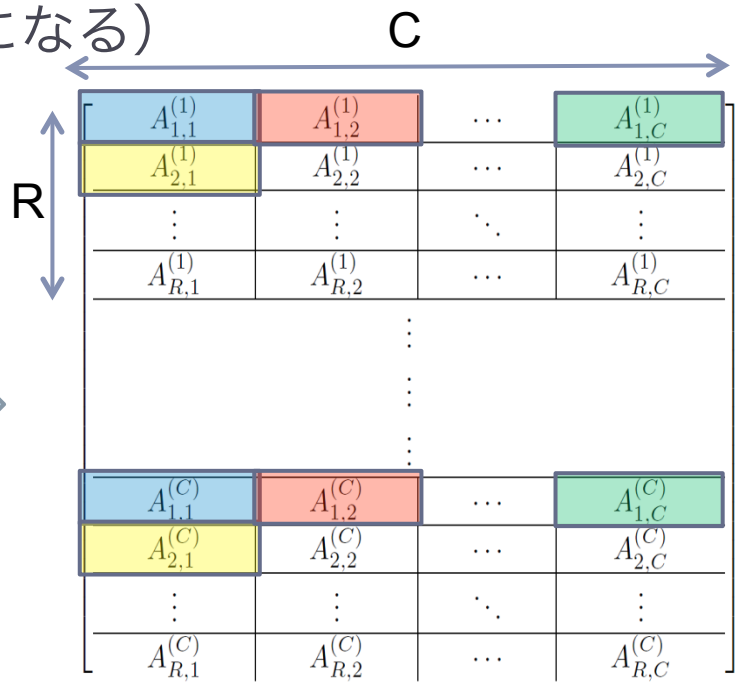
隣接行列の２次元分割※

※Andy Yoo, et. all, *A Scalable Distributed Parallel Breadth-First Search Algorithm on BlueGene/L.* SC05.

- 2次元分割BFSでは深さ1レベルの計算で2つの通信フェーズがある
  - Expand: Fronter頂点を**行方向**で共有
  - Fold: Neighbors情報を**列方向**で通信
  - （Bottom-upでは行と列が逆になる）



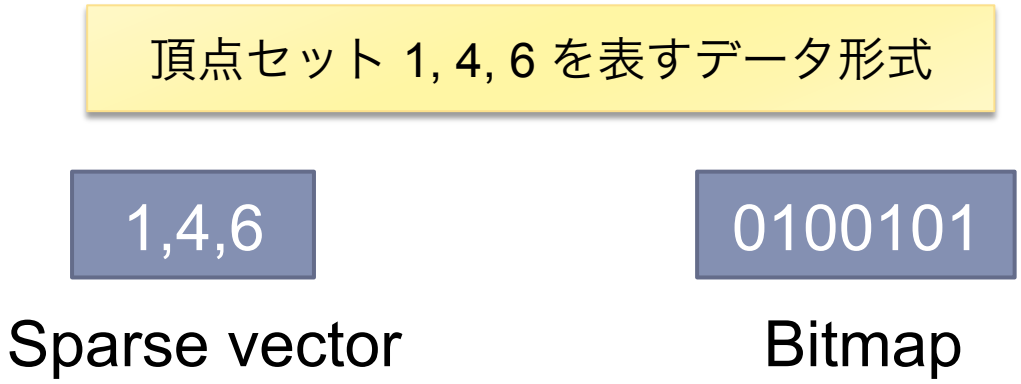
プロセッサの２次元配置



隣接行列の２次元分割

頂点濃度に応じたデータ形式選択による通信データ量削減

- BFSで通信するデータはFrontierの頂点や未訪問頂点
- 主に２種類のデータ形式が考えられる



- Frontierや未訪問頂点数が多い＝頂点濃度が高い
  - Bitmapの方が有利
- Frontierや未訪問頂点数が少ない＝頂点濃度が低い
  - Sparse vectorの方が有利

頂点濃度に応じたデータ形式選択による通信データ量削減

- トップダウンやボトムアップフェーズにこだわらず、頂点濃度によってBitmap と Sparse vectorから最適なデータ形式を選択

Graph500グラフにおける典型的なBFS探索におけるデータ形式選択

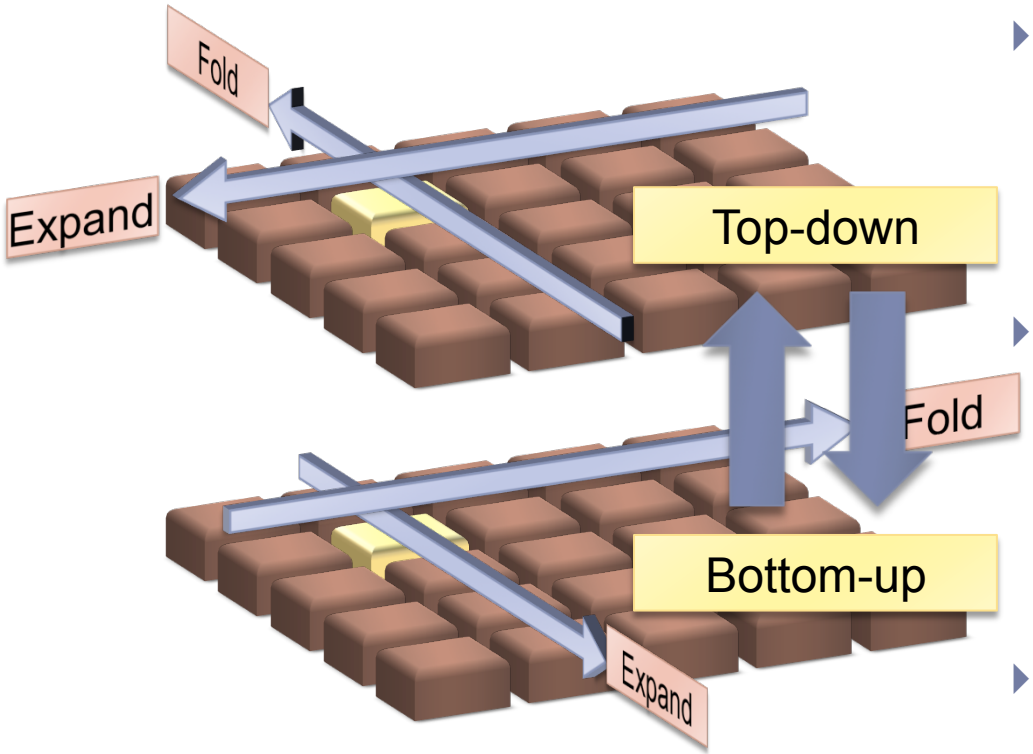
Level	Expand	Fold	Direction
1	Sparse vector	Sparse vector	top-down
2	Sparse vector	Sparse vector	top-down
3	Sparse vector	Bitmap	bottom-up
4	Bitmap	Bitmap	bottom-up
5	Bitmap	Sparse vector	bottom-up
6	Sparse vector	Sparse vector	top-down
7	Sparse vector	Sparse vector	top-down
8	Sparse vector	Sparse vector	top-down

Level5では、フロンティア頂点が全頂点の84.5%なので、bottom-upが有利だが、未訪問の頂点はわずか2.4%なので、Sparse vectorで処理したほうがいい

Level 5における通信データ量比較

Bitmap	Sparse vector
32 MB	**4.5 MB**

Efficient Hybrid Search with 2D Partitioning



: Graph data owned by a single processor

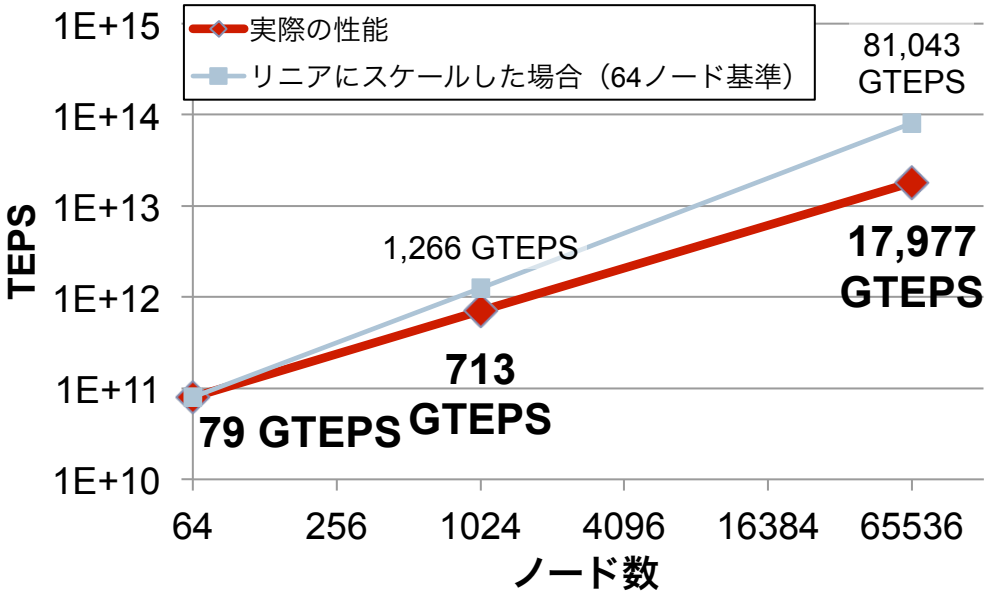
Sharing the same graph data between two direction search.

- Our approach is based on the Top-down and bottom-up hybrid search BFS. [Beamer2011, 2012]
- We realized the hybrid search without any increase of memory footprint.
  - Graph data is shared between two search directions.
- Overlapped communication with computation.
  - Both top-down and bottom-up utilize overlapping communication.

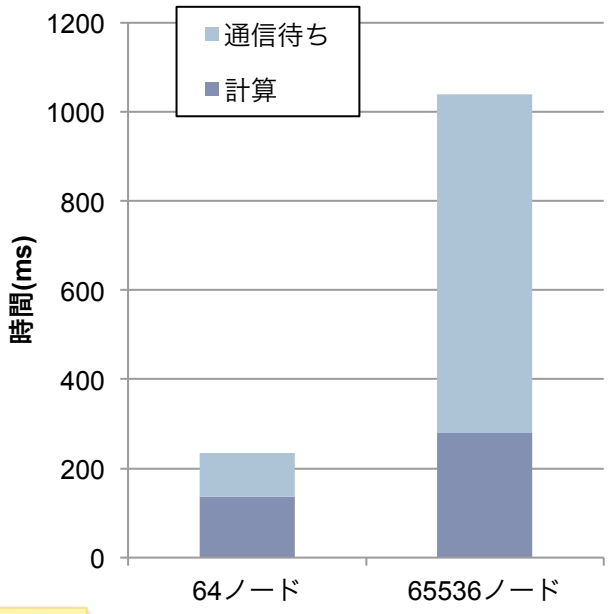
「京」における性能評価

- Graph500ベンチマークで用いるグラフを使用
- 65,536ノードで17,997GTEPSを達成
  - ただし、通信時間の増加はまだ最適化の余地あり
- 各最適化の効果の検証は今後の課題とする

ウィークスケーリング



ブレークダウン



問題サイズは65536ノードでScale40

The Graph500

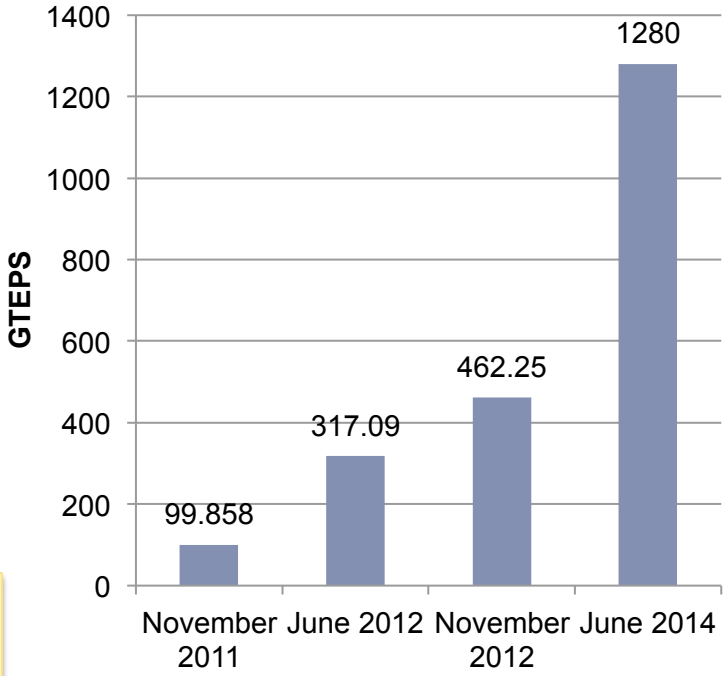
K Computer and TSUBAME 2.0 & 2.5

Graph500 ranking history for TSUBAME2.0 and 2.5

List	Rank	GTEPS	Implementation
November 2011	4	99.858	Top-down only
November 2012	20	462.25	GPU
June 2014	12	1280	**Efficient hybrid**
November 2014	??	1345	**Efficient hybrid**

*Every score is obtained using TSUBAME2.0 1366 nodes or TSUBAME 2.5 1024 nodes

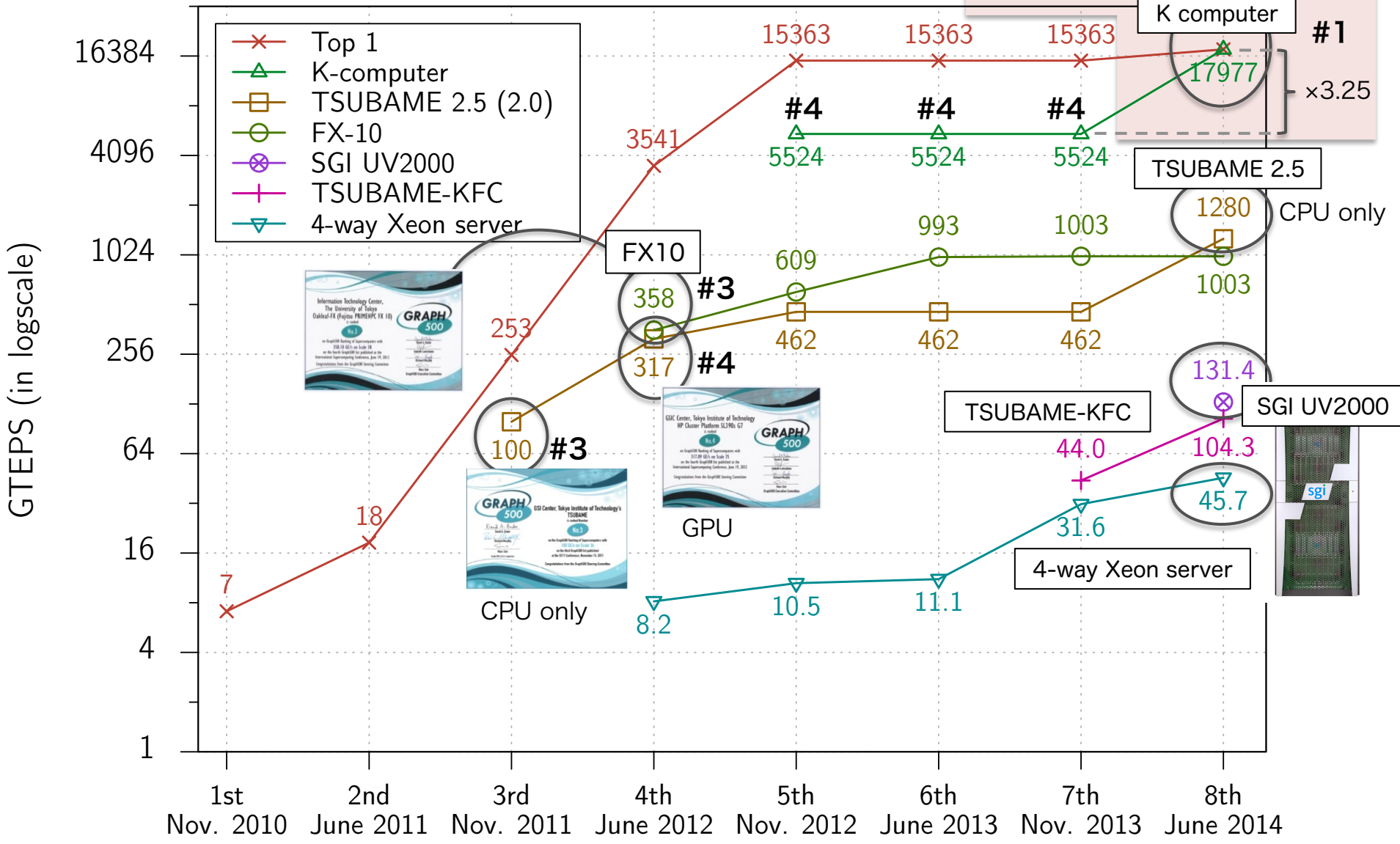
BFS performance on TSUBAME2.0 and 2.5



Graph500 ranking history for K Computer

List	Rank	GTEPS	Implementation
November 2013	4	5524.12	Top-down only
June 2014	1	17977.05	**Efficient hybrid**

Graph500ベンチマークにおける当CRESTチームの成果



RIKEN Advanced Institute for Computational Science (AICS)'s K computer

is ranked

No.1

on the Graph500 Ranking of Supercomputers with

17977.1 GE/s on Scale 40

on the 8th Graph500 list published at the International Supercomputing Conference, June 22, 2014.

Congratulations from the Graph500 Executive Committee

GRAPH 500

David A. Bader

Andrew Lumsdaine

Richard Murphy

Marc Snir

Graph500 Executive Committee

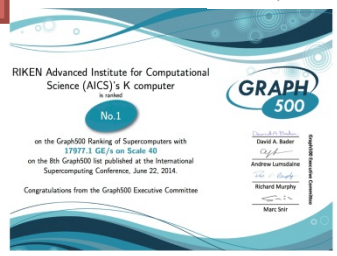
The Graph500 List in June 2014

http://www.graph500.org

- Measures performance using **TEPS** (# of Traversed edges per second) in graph traversal such as **BFS**

	No.	Rank	Machine	Installation Site	Number of nodes	Number of cores	Problem scale	GTEPS
Distributed Memory	1	1	K computer (Fujitsu - Custom supercomputer)	RIKEN Advanced Institute for Computational Science (AICS)	65536	524288	40	17977.1
Distributed Memory	12	12	TSUBAME 2.5 (HP - Cluster Platform SL390s G7)	Global Scientific Information and Computing Center, Tokyo Institute of Technology	1024	12288	36	1280.43
Shared Memory	42	42	ismuv2k2 (SGI - SGI UV 2000)	The Institute of Statistical Mathematics	1	640	32	131.427
Distributed Memory	46	46	TSUBAME-KFC (NEC - self-made)	Tokyo Institute of Technology	32	384	32	104.31
Shared Memory	52	52	GraphCREST-Sandybridge-EP-2.7GHz (HPCTECH Corporation - Intel(R) Xeon(R) CPU E5-4650 @ 2.70GHz (4 sockets))	Kyushu University	1	32	27	45.71

Fastest of multi-node



Fastest of single-node

Fastest of single-server

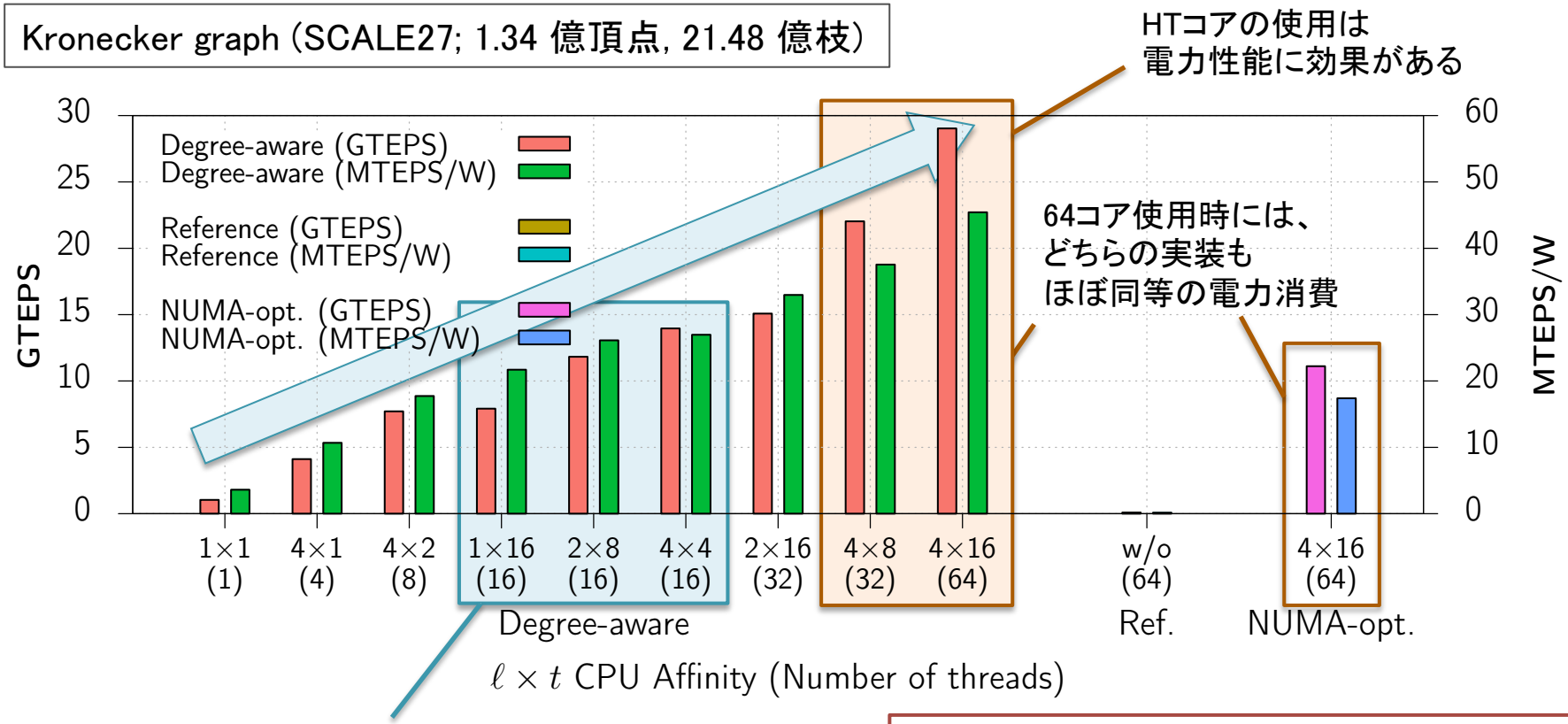
BFS 性能　　BFS 電力性能

アフィニティ設定毎の GTEPS と MTEPS/W

4-way Intel Xeon E5-4640 2.4GHz 上での数値実験

- BFS 性能 GTEPS と BFS 電力性能 MTEPS/W に共通する性質
  - 同じ**スレッド数**ならば、**ソケット数**は多いほうが良い
  - 同じ**ソケット数**ならば、**スレッド数**は多いほうが良い

Kronecker graph (SCALE27; 1.34 億頂点, 21.48 億枝)



使用するソケット数を増やすと電力消費も増えるが、
それ以上の TEPS の向上により TEPS/W も改善する

64 スレッド (4-socket x 16-threads) で
29.0 GTEPS と **45.43 MTEPS/W** を達成

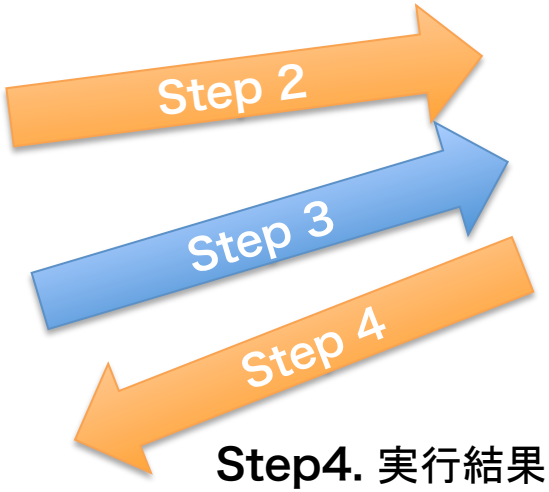
Android OS 上での数値実験

- Android NDK (Native Development Kit) の利用
  - オーバヘッドの少ない C/C++ で実装可能（Java は速度的に不向き）
  - GCC をベースとしたクロス・コンパイラの提供
  - OpenMP によるスレッド並列計算 &高速な GCC 拡張のアトミック関数
- Android Developer Tools によるログイン・転送

スマートフォン
SONY Xperia-A-SO-04E
CPU : 4-core Snapdragon
RAM : 2 GB

Step1. Android NDK による
クロスコンパイル
CC=arm-linux-androideabi-gcc

Step2. バイナリの転送
“adb push” コマンド



Step3. Android Developer Tools
によるログイン、Shell による操作
“adb shell” コマンド

Step4. 実行結果の取得
“adb pull” コマンド

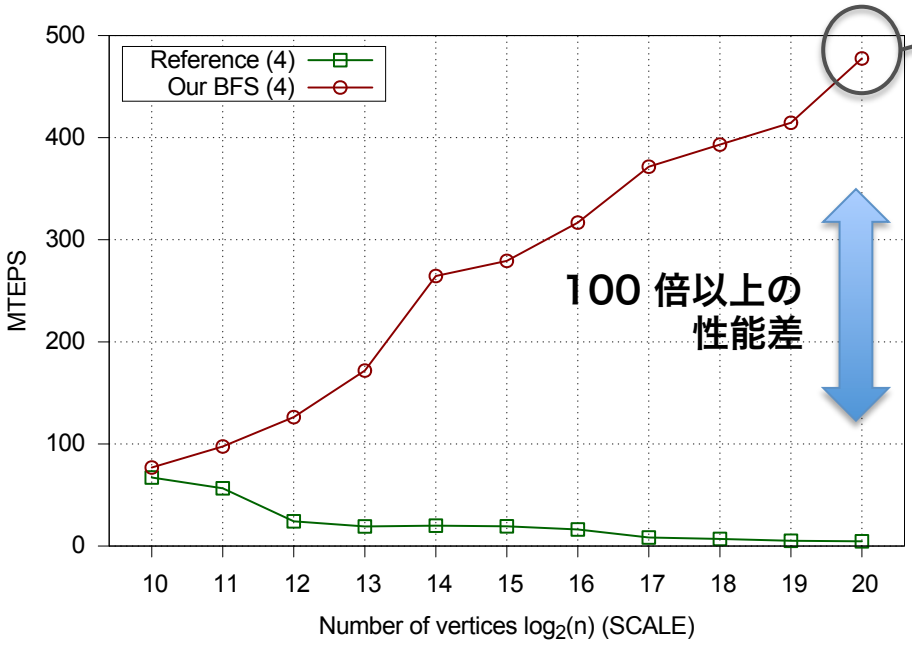
Android Developers: http://developer.android.com.

Xperia-A-SO-04E 上での BFS 電力性能

- **高速 (高性能) な実装が電力消費も良い**

Android OS 端末でもある程度の規模まで扱える

1,048,576 (= 2^{20}) 点
16,777,216 (= 2^{24}) 枝



スマートフォン
SONY Xperia-A-SO-04E
CPU : 4-core Snapdragon
RAM : 2 GB

Implementation	SCALE	MTEPS	watt	MTEPS/W
Reference ($p = 1$)	20	3.25	3.15	1.03
Reference ($p = 4$)	20	4.58	3.22	1.42
This study ($p = 1$)	20	136.29	3.23	42.25
This study ($p = 2$)	20	248.08	2.99	82.92
This study ($p = 4$)	20	477.63	3.12	153.17

**使用スレッド数と
電力消費の依存性は低い**

The Green Graph500 List in June 2014

http://green.graph500.org

- Measures power-efficiency using TEPS/W ratio

Intel Xeon E5-4640
59.12 MTEPS/W (28.48 GTEPS)

SONY Xperia-Z1-SO-01F
235 MTEPS/W (1.03 GTEPS)

NVIDIA JetsonTK1
204 MTEPS/W (1.63 GTEPS)

Big Data category

Rank	MTEPS/W	Site	Machine	G500 rank	Scale	GTEPS	Nodes
1	**59.12**	Kyushu University	GraphCREST-SandybridgeEP-2.4GHz		30	28.48	1
2	**48.29**	Kyushu University	GraphCREST-Sandybridge-EP-2.7GHz		30	31.95	1
3	**35.21**	Tokyo Institute of Technology	GraphCREST-Custom #1		31	13.8	1
4	**28.88**	Tokyo Institute of Technology	MEM-CREST Node #2		30	7.98	1
5	**17.24**	Kyushu University	GraphCREST-Bulldozer		31	13.63	1
6	**14.06**	Tokyo Institute of Technology	TSUBAME-KFC		32	104.31	32
7	**12.48**	The Institute of Statistical Mathematics	ismuv2k2		32	131.43	1

Small Data category

Rank	MTEPS/W	Site	Machine	G500 rank	Scale	GTEPS	Nodes
2	**235.15**	Kyushu University	GraphCREST-Xperia-Z1-SO-01F		20	1.03	1
3	**230.41**	Kyushu University	GraphCREST-Xperia-A-SO-04E		20	0.74	1
4	**204.38**	Tokyo Institute of Technology	EBD-GoldenBox-Prototype		21	1.64	1
5	**180.76**	Kyushu University	GraphCREST-Xperia-A-SO-04E		21	0.59	1
6	**171.77**	Kyushu University	GraphCREST-Xperia-Z1-SO-01F		21	0.91	1
7	**153.17**	Chuo University	GraphCREST-Xperia-A-SO-04E	143	20	0.478	1

グラフ解析と最適化の今後

九州大学
「センター・オブ・イノベーション（COI）プログラム」

共進化社会システム創成拠点

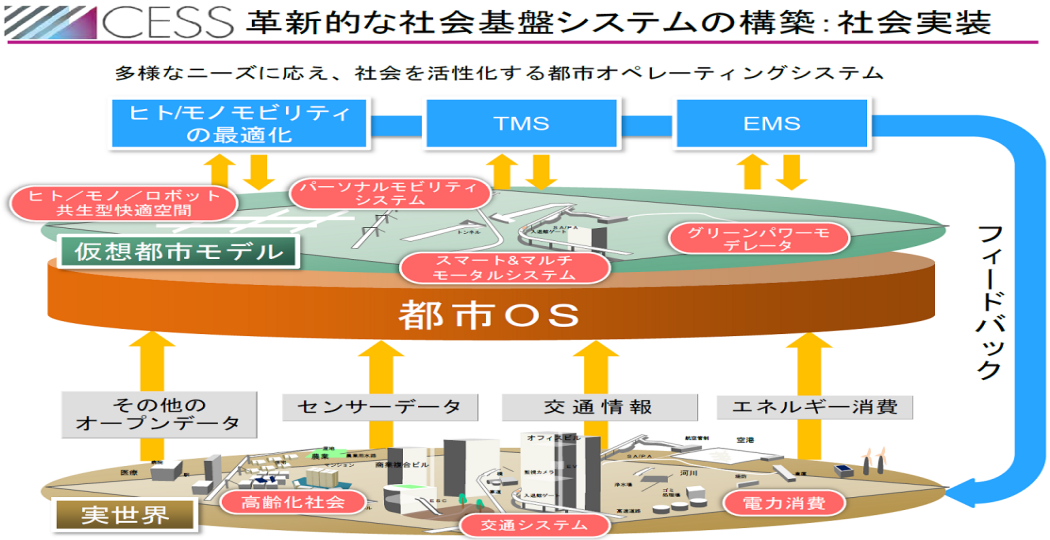
CESS
Center for Co-Evolutional Social System
共進化社会システム創成拠点

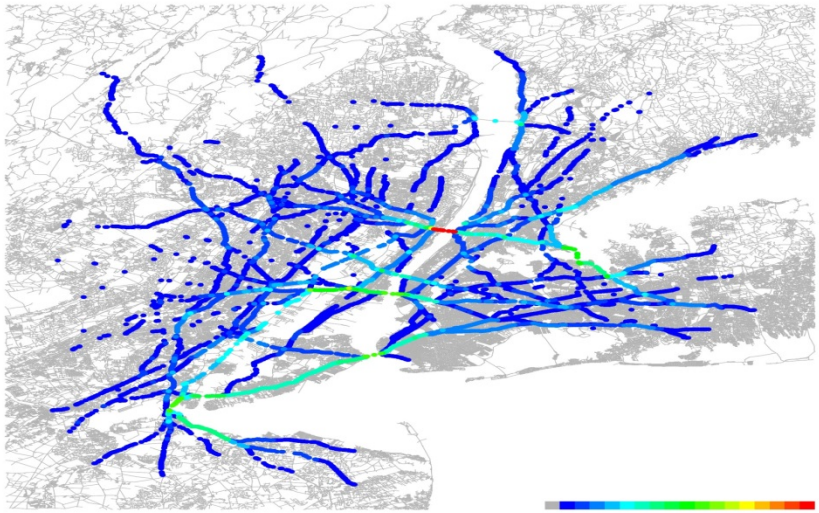
KYUSHU UNIVERSITY

COI STREAM

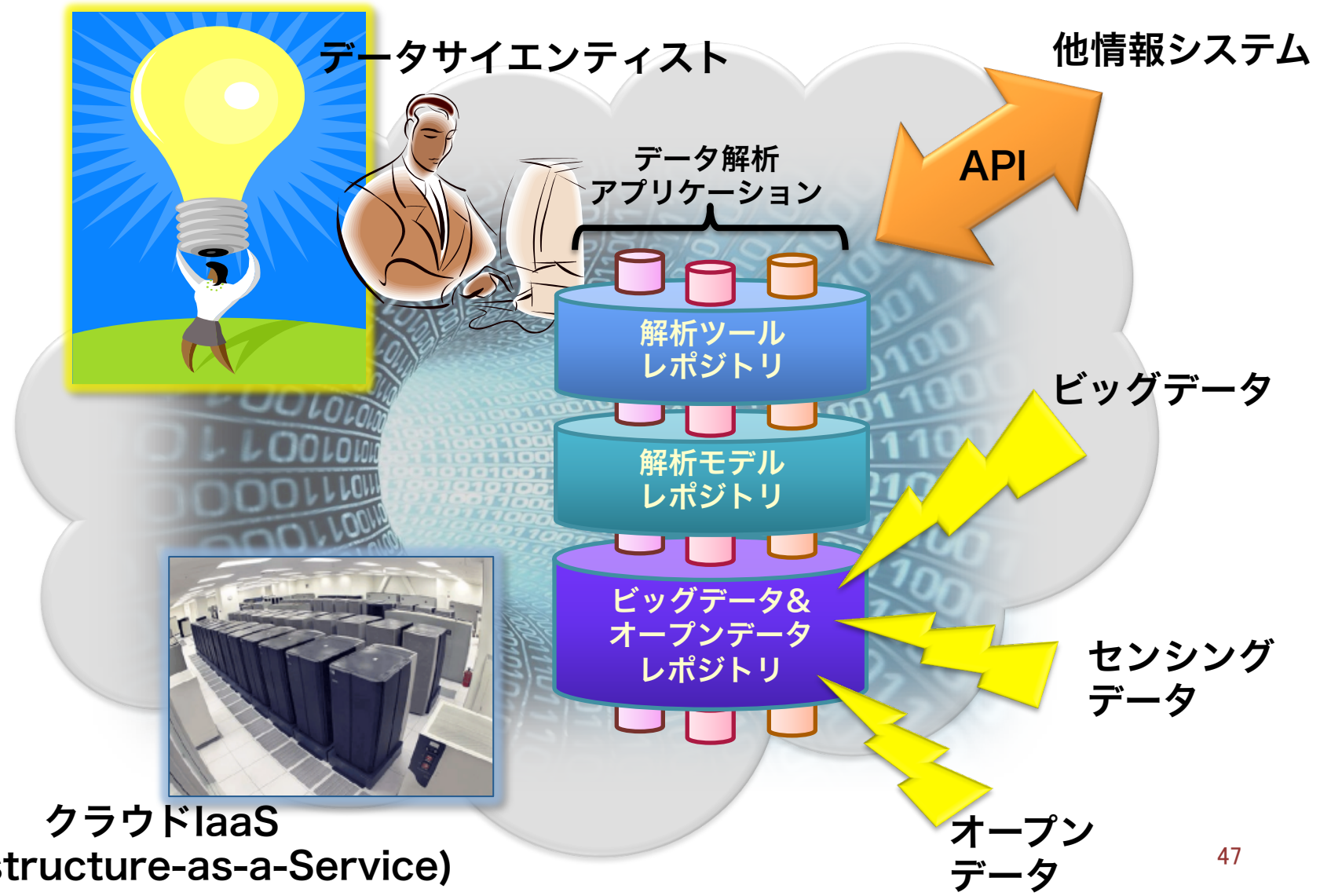
実問題に対する大規模グラフ解析

- 超大規模ネットワークに対する探索アルゴリズムとクラスタリングアルゴリズムの高速実装
- 数百万頂点～数兆頂点、数億枝～数百兆枝からなる超大規模なグラフ解析
  - 数百万人の被災者の避難経路の計算では数千万頂点のグラフ(１スレッドあたりのメモリ要求量1Gbytes)に対して、同時に数百万スレッド単位で各被災者毎の最短路計算と各点の重要度判定
- **新しい産業応用の開拓** → １：ヒト/モノ　２：エネルギー　３：情報のモビリティ→ **数理モデルとスパコンを用いた最適化 (整数計画問題に対する並列ソルバー等)**
  - 交通データに対する経路探索動的に変化する交通量等から高速な重要度判定を行い、交通管制等に活用
  - ソーシャルネットワークデータに対する動的な重要度、影響度の判定各点の周辺、および広域内における影響（情報の伝播力）を推定





BODIC.org
(BigData & OpenData in the Cloud)



「ビッグデータ&オープンデータを活用した創業」支援策 [20140729PR]

1. 福岡市が有するビッグデータ&オープンデータ活用基盤

BODIK (ビッグデータ&オープンデータ研究会 in 九州)
[福岡市、URC、ISITが共同で昨年12月に発足]

DSS4K (データサイエンススクール for 九州)
[ISITが企画中]

BODIC.org (BigData & OpenData in the Cloud)
[ISIT&九大共同開発中、本年10月からISITが運用開始予定]

データ解析ツール

データサイエンティスト

データ貯蔵庫

センサーネットワーク

福岡市オープンデータサイト [10月開設予定]

業務データ

3. 人流センシングデータ活用で可能となる多様な創業

[例1] 店舗向けリアルタイムマーケッティングサービス&顧客向け店舗ナビゲーションサービスビジネス

人流解析

顧客の購買履歴や好み、店舗の品揃え、等に基づいたパーソナル店舗ナビゲーション

周辺顧客の属性、購買履歴や好み、動線、等に基づいたリアルタイムマーケッティング情報提供

[例2] スマーターモバイル"XXX"サービスビジネス

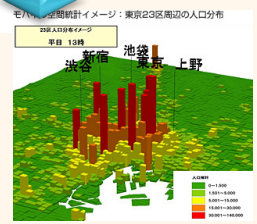
人流解析

人流に応じて様々な移動型(モバイル)サービスを提供

2. 規制緩和でもっと容易になる「人流センシング」

[現状] 携帯電話キャリア各社による位置情報収集&空間統計情報提供

基地局で収集した携帯端末の位置情報を基に空間統計情報を作成



例) NTTドコモ モバイル空間統計

3つの課題
1. キャリアが情報独占
2. リアルタイムでの情報提供は未実施
3. 元データへのアクセス不可(統計情報のみ利用可)

規制緩和

[規制緩和後] Wi-Fiアクセスポイントを用いた位置情報収集&BODIC.org経由のリアルタイムな元データ提供

Wi-Fiアクセスポイントは、技術的には周辺の携帯端末の識別情報(MACアドレス)が収集可能。しかし、現在は未活用(利用不可)

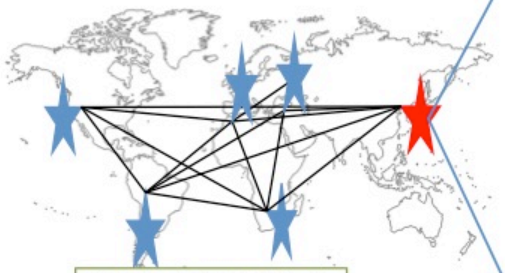
2つの規制緩和案
1. Wi-Fiアクセスポイントでの携帯端末識別情報収集を許可
2. 携帯端末所有者の合意の下、収集した識別情報の元データの2次利用を可能に

※九大伊都で実証実験中

都市OSの機能

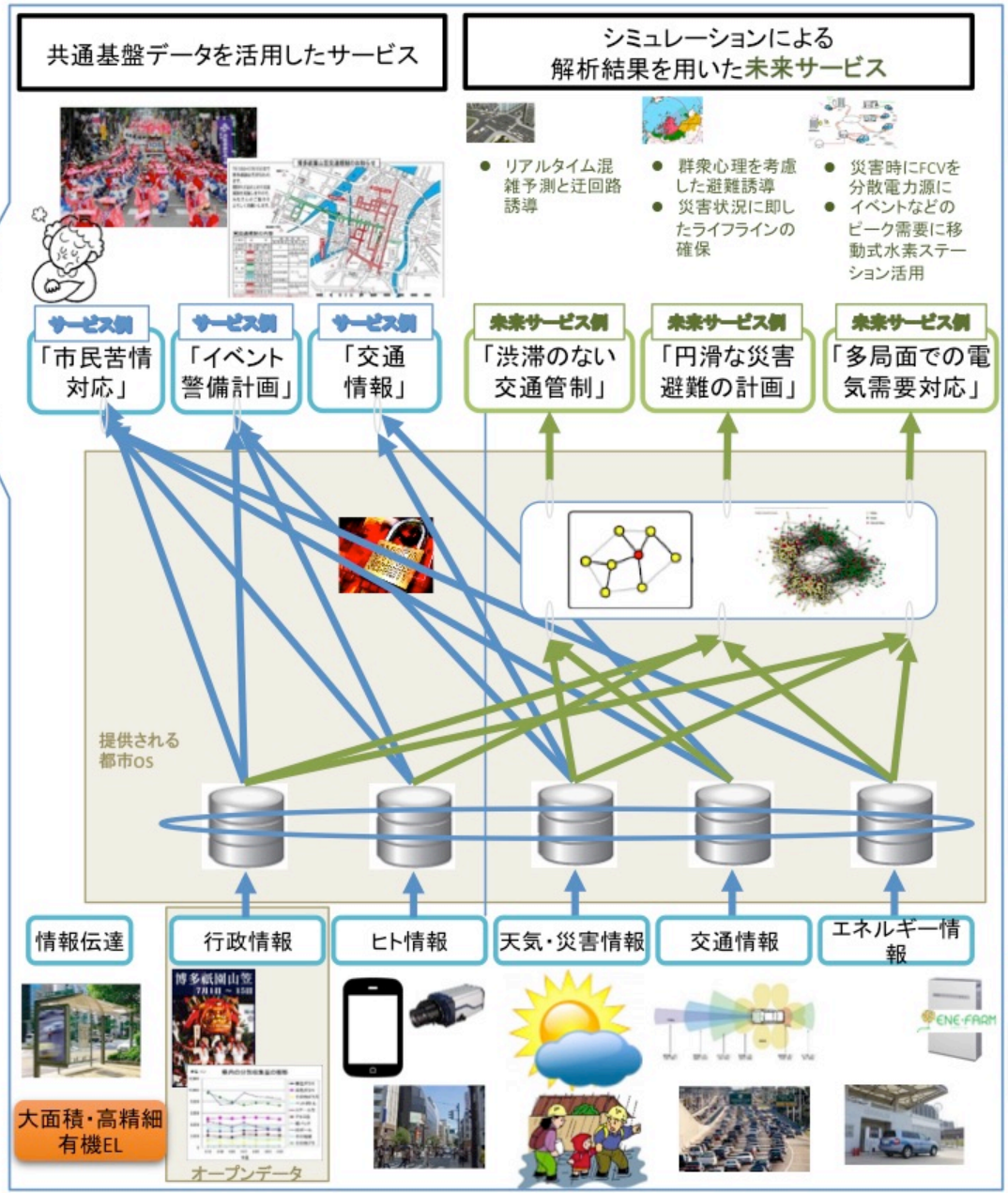
都市OSの展開

複数の都市への展開
複数の都市でデータ共有



世界の各都市を結ぶ
都市OSネットワーク

■ 都市OSの機能
- 新しい都市サービスの共通基盤の提供
- 都市間のデータやサービス共有の仕組み
- 都市規模に応じたスケーラビリティの担保
- 災害時などの緊急時サービスの他都市での代替
- 新しいサービス・アプリケーションのための抽象化



アプリケーション・サービス

共通データを様々な市民サービスへ展開
- 新規民間サービス支援のしくみ
- エネルギー可視化システム
- 最適なHEMSと制度
- スマート＆マルチモーダル交通

起業 創業

最適化・分析

自動で最適化、制御
ボトルネック検出
- リアルタイムシミュレーション
- セキュリティ

データの格納

いろいろな情報を共通データ基盤に吸い上げ、必要に応じて取り出し
- 都市OSコアソフトウェア
- データ格納リポジトリ

データ

オープンデータ
センサーネットワーク
- 経済的・快適な燃料電池車
- 経済的・エコな燃料電池

グラフ探索及び数理最適化ライブラリによる大規模グラフ処理

大

計算量

小

小

データ量

大

上層：NP 困難な最適化アルゴリズムの実行

1. **MIP(混合整数計画問題)**の場合：0-1 整数変数の数 = n で計算量はおよそ $O(2^n)$
2. 前処理による変数の削除と並列計算の適用（CPUコア中心の大規模スレッド並列：分枝カット法の適用）
3. **データ量は 10^5 以下（整数変数の数）**
4. 施設配置問題、集合被覆（分割）問題、スケジューリング問題などの最適化問題

中層：多項式時間最適化アルゴリズムの実行

1. **SDP(半正定値計画問題)**の場合：n = 行列の大きさ, m = 制約式の数で計算量はおよそ $O(n^3 + m^3)$
2. 疎性の追求と前処理さらに並列計算の適用（CPUコア中心の大規模スレッド並列が中心だが、今後は CPU + GPU による高速化）
3. **データ量は 10^8 以下(非負変数の数)：10^6 以下(制約条件数)**
4. グラフ分割、センサーネットワーク、サポートベクターマシンなどの最適化問題

下層：グラフ解析アルゴリズムの実行

1. **ダイクストラ法**(1対全最短路問題：ヒープ付き)の場合：n = 点数, m = 枝数で計算量はおよそ $O((n + m)\log n)$
2. グラフ探索の局所的な評価では優先キュー(ヒープ木)を用いる⇒実行時間、メモリ消費量が安定的（CPUコア中心の大規模スレッド並列が中心だが、高速ストレージ技術による超大規模グラフ処理）
3. **データ量は $10^{12\sim14}$ 以下（グラフの点数と枝数）**
4. 最短路計算、ネットワーク内での各点の重要度を推定。各点の周辺、及び広域内における影響（情報の伝播力）を計算

下層：グラフ解析アルゴリズムの実行

1.ダイクストラ法(1対全最短路問題：ヒープ付き)の場合：n＝点数, m＝枝数で計算量 O((n＋m)log n)

2. グラフ探索の局所的な評価では優先キュー(ヒープ木)を用いる⇒実行時間、メモリ消費量が安定的（CPUコア中心の大規模スレッド並列が中心＆高速ストレージ技術による超大規模グラフ処理)

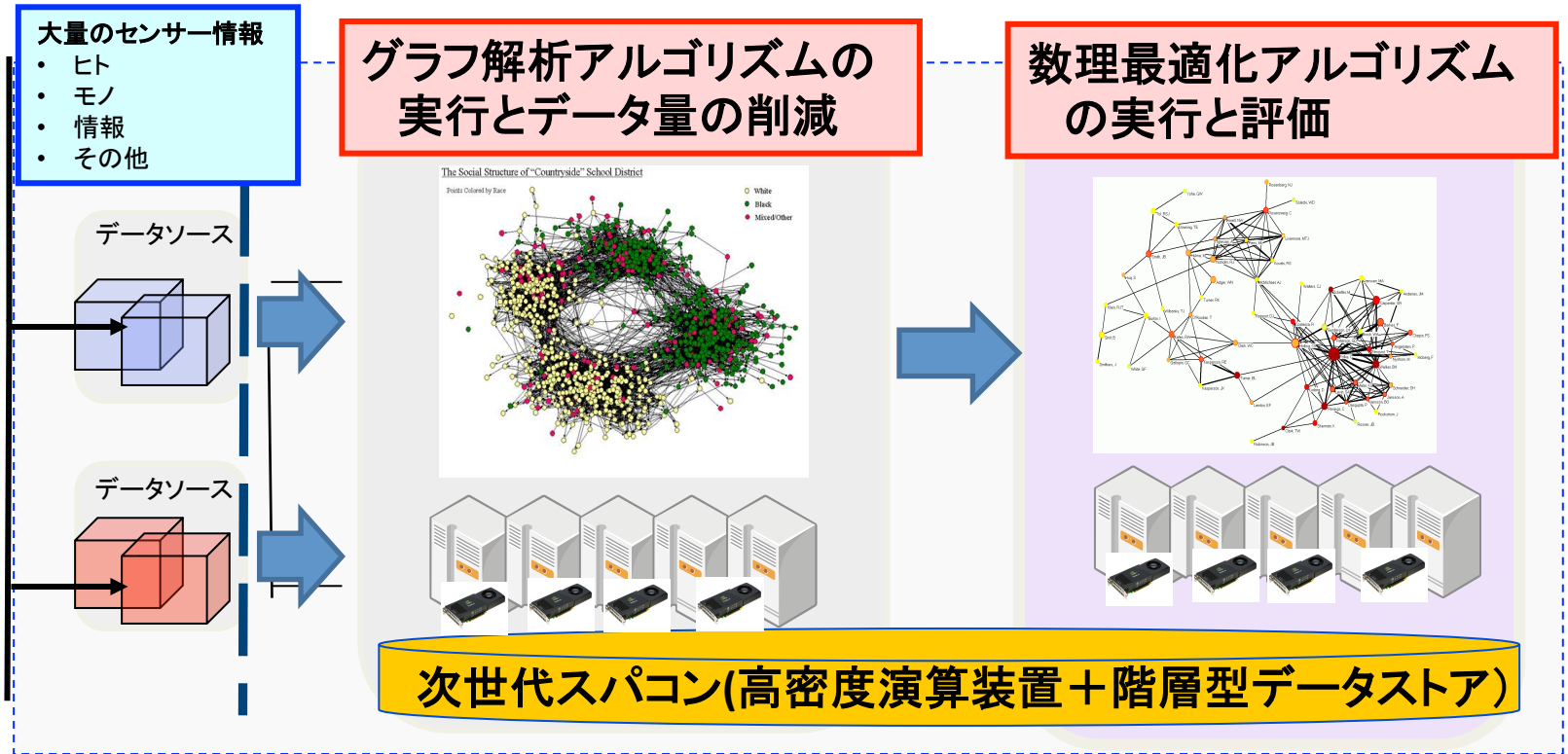
3.データ量は $10^{12\sim14}$ 以下（グラフの点数と枝数)

4. 最短路計算、ネットワーク内での各点の重要度を推定。各点の周辺、及び広域内における影響（情報の伝播力)を計算

- BFS(幅優先探索) → Graph500 : 京コンピュータ上で Scale 40 (2^{40}点, 2^{44} 枝）のグラフに対して 18 TeraTEPS の性能 (Top-down & Bottom-up & Level Synchronized BFS)
- Shortest Path (最短路) → IBM BG/Q 上で Scale 38 (2^{38}点, 2^{42}枝）のグラフに対して 3.1TeraTEPSの性能(Delta Stepping & Bellman Ford のハイブリッド) → 静的なグラフに対してはHighway hierarchies などの前処理が有効
- Maximum Flow (最大フロー) → Push Relabel 法の実装 → 最速フローに適用 →約 4,588万点, 18,537万枝のグラフに対して最大フローの 4～5分の実行時間
- Centrality (中心性) → 前処理や並列計算等も有効
  - 各点に接続されている枝数（Degree Centrality）
  - 各点が最短路に含まれる回数（Betweenness Centrality）
  - 最も遠い点までの最短距離（Graph Centrality）
  - 各点までの最短距離（Closeness Centrality）
- その他のアルゴリズムとツールも多数存在
  - Minimum spanning tree ： 全域木
  - Strongly connected component ： 強連結成分
  - Spectral clustering ： グラフの行列表現(Laplacian Matrix) → 固有値計算 → クラスタリング
  - Hyper ANF : Neighborhood 関数の近似 ： 点 x から t hop で到達する点 y → (x,y) の計算

グラフ探索及び数理最適化ライブラリによる大規模グラフ処理基盤

- 大規模センサーから到着するストリーミングデータに対して精緻な解析を実現する大規模グラフ処理基盤を開発する
- 大規模グラフ処理基盤は以下の処理系から構成される
  - グラフ解析アルゴリズムの実行： 最短路計算、ネットワーク内での各点の重要度を推定。各点の周辺、及び広域内における影響（情報の伝播力)を計算 → 重要な要素を失うことなくデータ量を削減する
  - 数理最適化アルゴリズムの実行： 施設配置問題、集合被覆（分割）問題、スケジューリング、配送計画問題などの数理最適化問題 → モビリティに関する改善提案
  - 大規模なセンサーデータを高速かつ重要性を失うことなく縮約することよって、精緻な解析を実現
- 計算及びデータ蓄積の基盤となる次世代スパコン
  - 並列数の爆発的増大、不均質化、高密度化 ＆ 記憶装置の多階層化・大容量化



数理計画問題(最適化問題)と2015年予想(目標)

- **非常に応用が広範(企業、社会、公共政策)**→ 高性能なソルバーを作ること自体が最適化問題
- センサーデータによる最適化問題の複雑＆巨大化
- **半正定計画問題(SDP)**と**混合整数計画問題(MIP)**が2大注目数理計画問題
- **汎用ソルバーの必要性**(個別の問題に対する仮定やチューニングは効果が低い)
- 入力は**疎データ**と**密データ**の混合
  - **疎データ**:多数のCPUコアによる処理(浮動小数点演算性能に非依存:密データへ変換)→ **GPU 系による処理**
  - **密データ:GPU 系による処理**
  - 理論的性能限界等からボトルネック箇所を特定
  - 数値演算能力とメモリバンド等のトレードオフ関係を把握
  - 計算量とデータ移動量の正確な推定、疎性やサイズなどのデータ特性と性能値の見極め
- 計算量とポストペタスパコンでの想定:
  - **計算量$O(n^3)$ → 数百万程度, $O(n \log n)$ → 100億以上, $O(n)$ → 100兆以上**