最新の気象予測
～その理論と技術～

北海道大学地球惑星科学集中講義
札幌管区気象台　室井ちあし

集中講義の内容

- １日目
  - 気象予測の基礎
  - 気象とコンピュータ
  - 数値予報モデル
- ２日目
  - 数値予報モデル（続）
  - データ同化
  - アンサンブル予報
  - （セミナー）北海道の気象災害リスクと防災気象情報

気象とコンピュータ

北海道大学地球惑星科学集中講義
札幌管区気象台　室井ちあし

はじめに

- 理論、実験を並び、シミュレーションは「第3の科学」
- 近年は、データ解析は「第4の科学」と呼ばれる
- 実験が困難な現象の解明や実験に時間や費用がかかりすぎる場合に、仮想的な実験手段として用いられる
- 気象分野はとりわけ、シミュレーション、データ解析が盛んで、親和性が高い
- 天気予報以外にも、気象研究に広く活用されている

気象分野でのコンピュータの利用

- 予測モデル
  - 計算が複雑で量も多く、高速なコンピュータ、スーパーコンピュータが利用されることが多い
- データ解析
  - データ取扱量が多く、巨大なストレージ（記憶装置）が利用されることが多い
- 可視化

予測モデル

予測モデルを用いた研究パターン

- 「できあい」の予測モデルを移植・利用して、結果を出す
  - 事例解析
- モデルの数値計算に関する研究
- 物理プロセスに関する研究

日本の全球モデル

- 気象庁GSM
  - 天気予報～気候予測ほか、様々な業務の基盤
- 東大・JAMSTEC　MIROC
- 電脳倶楽部

- 東大・JAMSTEC　NICAM
  - 正二十面体型全球雲解像モデル
- JAMSTEC　MESG
  - インヤン格子型全球・領域モデル

日本のメソモデル

- 気象庁 NHM
  - 気象研究所、気象庁で開発、研究でも広く利用されている
- 名大 CReSS
  - 「地球シミュレータ」を契機に、雲解像モデルとして開発されている
- 電脳倶楽部
- 気象庁 asuca
  - NHMに代わる新モデル、今年春から実用化

世界のメソモデル

- WRF
  - 米国のコミュニティモデル
- ARPS
  - 米国オクラホマ大学で開発されたモデル
- MM5
  - 米国NCARで開発されていたモデル
- COSMO
  - ヨーロッパのコミュニティモデル

予測モデル研究の課題

- 「できあい」のモデル
  - 効果的に結果が出せる一方、現代のモデルは巨大化していて、ブラックボックス化
  - 結果が正しいのかの検討が不十分になりがち


スーパーコンピュータ

スーパーコンピュータ

- 普通のコンピュータと何が違うのか?
- 原理は同じ
- クルマで言うところの「F1カー」

- 「京」はスーパーコンピュータの中のスーパーコンピュータ　→　事業仕分けの対象に

TOP500

- スパコン間の性能比較ランキングで最も有名なもの
- 年に2度、更新・発表される
- Linpack（大規模密行列の直接法による解法プログラム）を性能指標とする
  - 長時間負荷をかけ耐久性が示せる、歴史的な価値がある一方、1つのプログラムで性能を示すと言えるのか？という問題もある
- 消費電力といった環境を重視した“Green500” も別途存在し、年々競争が激しさを増している

スーパーコンピュータ性能ランキングの変遷

○今回1位となった中国・国防科学技術大学(NUDT)のTianhe-2(天河2号)は、LINPACK実効性能33.86ペタFLOPS、実行効率61.6%。中国のスパコンが世界1位となるのは、2010年11月のTianhe-1A(天河1A号)以来、2回目。

○TOP10ランクイン状況としては、米国(5システム)が1位、中国及びドイツ(2システム)が2位、日本が1システム。TOP100ランクインでは、米国が47システムと圧倒的優位な状況となっている。次いで、日本(10システム)が2位、英国(8システム)が3位、中国及びフランス(6システム)が4位、ドイツ及びインド(4システム)が6位と続いている。

平成24年6月

順位	システム名称	サイト	ベンダ	国名	Linpack 演算性能 (テラFLOPS)
1	Sequoia	ローレンスリバモア研	IBM	米	16,325
2	「京」(K computer)	理研 計算科学研究機構(AICS)	富士通	日	10,510
3	Mira	アルゴンヌ研	IBM	米	8,162
4	SuperMUC	ライプニッツスーパーコンピューティングセンタ(LRZ)	IBM	独	2,897
5	Tianhe-1A(天河1A号)	天津スパコンセンタ	NUDT	中	2,566
6	Jaguar	オークリッジ研	Cray	米	1,941
7	Fermi	Cinecaコンピューティングセンター	IBM	伊	1,725
8	JuQUEEN	ユーリヒ総合研究機構(FZJ)	IBM	独	1,380
9	Curie thin nodes	フランス原子力庁	Bull SA	仏	1,359
10	Nebulae(星雲)	深圳スパコンセンタ	Dawning	中	1,271
12	Helios	国際核融合エネルギー研究センタ	Bull SA	日	1,237
14	TSUBAME2.0	東工大学術国際情報センタ(GSIC)	NEC/HP	日	1,192
18	Oakleaf-FX	東大情報基盤センタ	富士通	日	1,043
36	BlueGene/Q	高エネルギー加速器研究機構	IBM	日	518
41	HA-PACS	筑波大計算科学研究センタ	Appro/Cray	日	422
70	Hitachi SR16000	東北大学 金属材料研究所	日立	日	244
73	Camphor	京都大学	Cray	日	239
84	BX900	日本原子力研究開発機構(JAEA)	富士通	日	191

平成24年11月

順位	システム名称	サイト	ベンダ	国名	Linpack 演算性能 (テラFLOPS)
1	Titan	オークリッジ研	Cray	米	17,590
2	Sequoia	ローレンスリバモア研	IBM	米	16,325
3	「京」(K computer)	理研 計算科学研究機構(AICS)	富士通	日	10,510
4	Mira	アルゴンヌ研	IBM	米	8,162
5	JuQUEEN	ユーリヒ総合研究機構(FZJ)	IBM	独	4,141
6	SuperMUC	ライプニッツスーパーコンピューティングセンタ(LRZ)	IBM	独	2,897
7	Stampede	テキサス大学	Dell	米	2,660
8	Tianhe-1A(天河1A号)	天津スパコンセンタ	NUDT	中	2,566
9	Fermi	Cinecaコンピューティングセンター	IBM	伊	1,725
10	DARPA Trial Subset	DOD国防高等研究計画局 IBM開発セ	IBM	米	1,515
15	Helios	国際核融合エネルギー研究センタ	Bull SA	日	1,237
17	TSUBAME2.0	東工大学術国際情報センタ(GSIC)	NEC/HP	日	1,192
21	Oakleaf-FX	東大情報基盤センタ	富士通	日	1,043
39	SGI Altix X	電力中央研究所	SGI	日	582
41	HIMAWARI	高エネルギー加速器研究機構	IBM	日	518
42	SAKURA	高エネルギー加速器研究機構	IBM	日	518
45	PRIMERGY CX400	九州大学	富士通	日	460
51	HA-PACS	筑波大計算科学研究センタ	Appro/Cray	日	422
95	Hitachi SR16000	核融合科学研究所	日立	日	253
97	Camphor	京都大学	Cray	日	251.7
100	Hitachi SR16000	東北大学 金属材料研究所	日立	日	243.9

平成25年6月

順位	システム名称	サイト	ベンダ	国名	Linpack 演算性能 (テラFLOPS)
1	Tianhe-2(天河2号)	国防科学技術大学	NUDT	中	33,863
2	Titan	オークリッジ研	Cray	米	17,590
3	Sequoia	ローレンスリバモア研	IBM	米	17,173
4	「京」(K computer)	理研 計算科学研究機構(AICS)	富士通	日	10,510
5	Mira	アルゴンヌ研	IBM	米	8,587
6	Stampede	テキサス大学	Dell	米	5,168
7	JuQUEEN	ユーリヒ総合研究機構(FZJ)	IBM	独	5,009
8	Vulcan	ローレンスリバモア研	IBM	米	4,293
9	SuperMUC	ライプニッツスーパーコンピューティングセンタ(LRZ)	IBM	独	2,897
10	Tianhe-1A(天河1A号)	天津スパコンセンタ	NUDT	中	2,566
20	Helios	国際核融合エネルギー研究センタ	Bull SA	日	1,237
21	TSUBAME2.0	東工大学術国際情報センタ(GSIC)	NEC/HP	日	1,192
26	Oakleaf-FX	東大情報基盤センタ	富士通	日	1,043
43	PRIMERGY CX400	九州大学	富士通	日	621
47	SGI Altix X	電力中央研究所	SGI	日	582
50	SAKURA	高エネルギー加速器研究機構	IBM	日	537
51	HIMAWARI	高エネルギー加速器研究機構	IBM	日	537
62	HA-PACS	筑波大計算科学研究センタ	Appro/Cray	日	422
63	Cray XC30	国立天文台	Cray	日	420

気象庁 Japan Meteorological Agency

JMA

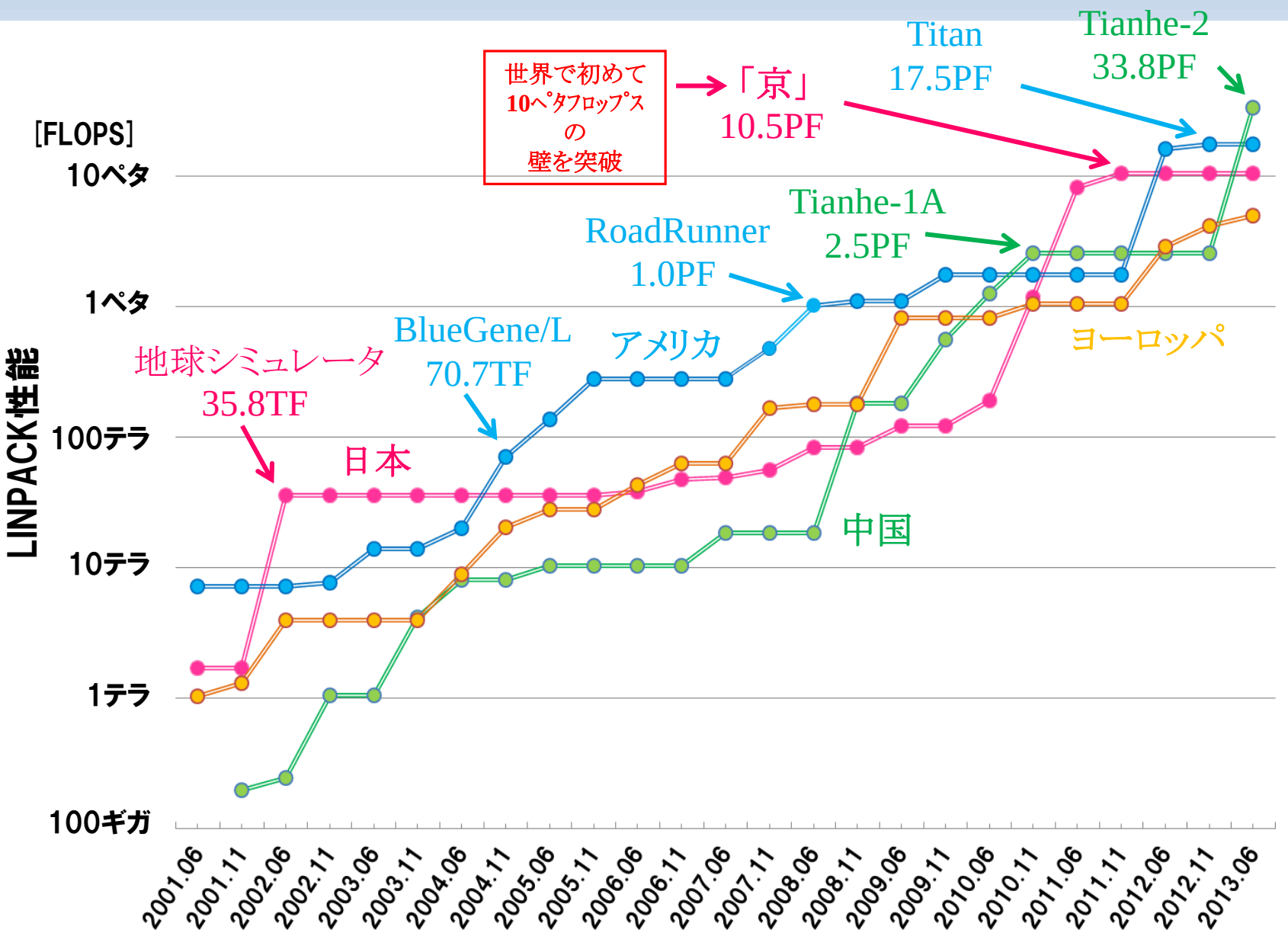
(文部科学省資料)

TOP500（平成25年6月）のうち日本に設置されているスパコン

順位	サイト	ベンダ	システム名称	Linpack 演算性能 （テラFLOPS）
4	理研　計算科学研究機構(AICS)	富士通	「京」（K computer）	10,510
20	国際核融合エネルギー研究センタ	Bull SA	Helios	1,237
21	東工大学術国際情報センタ（GSIC）	NEC/HP	TSUBAME 2.0	1,192
26	東大情報基盤センタ	富士通	Oakleaf-FX	1,043
43	九州大学	富士通	PRIMERGY CX400	621
47	電力中央研究所	SGI	SGI Altix X	582
50	高エネルギー加速器研究機構	IBM	SAKURA	537
51	高エネルギー加速器研究機構	IBM	HIMAWARI	537
62	筑波大計算科学研究センタ	Cray	HA-PACS	422
63	国立天文台	Cray	Cray XC30	420
118	核融合科学研究所	日立	Hitachi SR16000	253
121	京都大学	Cray	Camphor	252
124	東北大学 金属材料研究所	日立	Hitachi SR16000	244
129	分子科学研究所	富士通	PRIMERGY CX250 & RX300	238
138	サービスプロバイダ	HP	DL160 Gen8	227
161	日本原子力研究開発機構(JAEA)	富士通	PRIMERGY BX900	191
177	サービスプロバイダ	HP	BL460c Gen8	187
207	九州大学	富士通	PRIMEHPC FX10	167
212	東大 物性研	SGI	SGI Altix ICE 8400EX	162
273	エレクトロニクス関係	IBM	iDataPlex DX360M4	139
282	京都大学	Cray	Laurel	135
318	サービスプロバイダ	HP	DL360p Gen	126
334	金融関係	IBM	xSeries x3650M3	126
341	地球シミュレータセンタ	NEC	地球シミュレータ	122
344	北海道大学情報基盤センタ	日立	Hitachi SR16000 Model M1	122
393	JAXA	富士通	Fujitsu FX1	111
432	北陸先端科学技術大学院大学	Cray	Cray XC30	105
461	東大情報基盤センタ	日立	T2Kオープンスパコン	102
467	東大ヒトゲノム解析センタ	日立	HA8000-tc/HT225	101
480	理研情報基盤センタ(RIKEN)	富士通	RICC	98

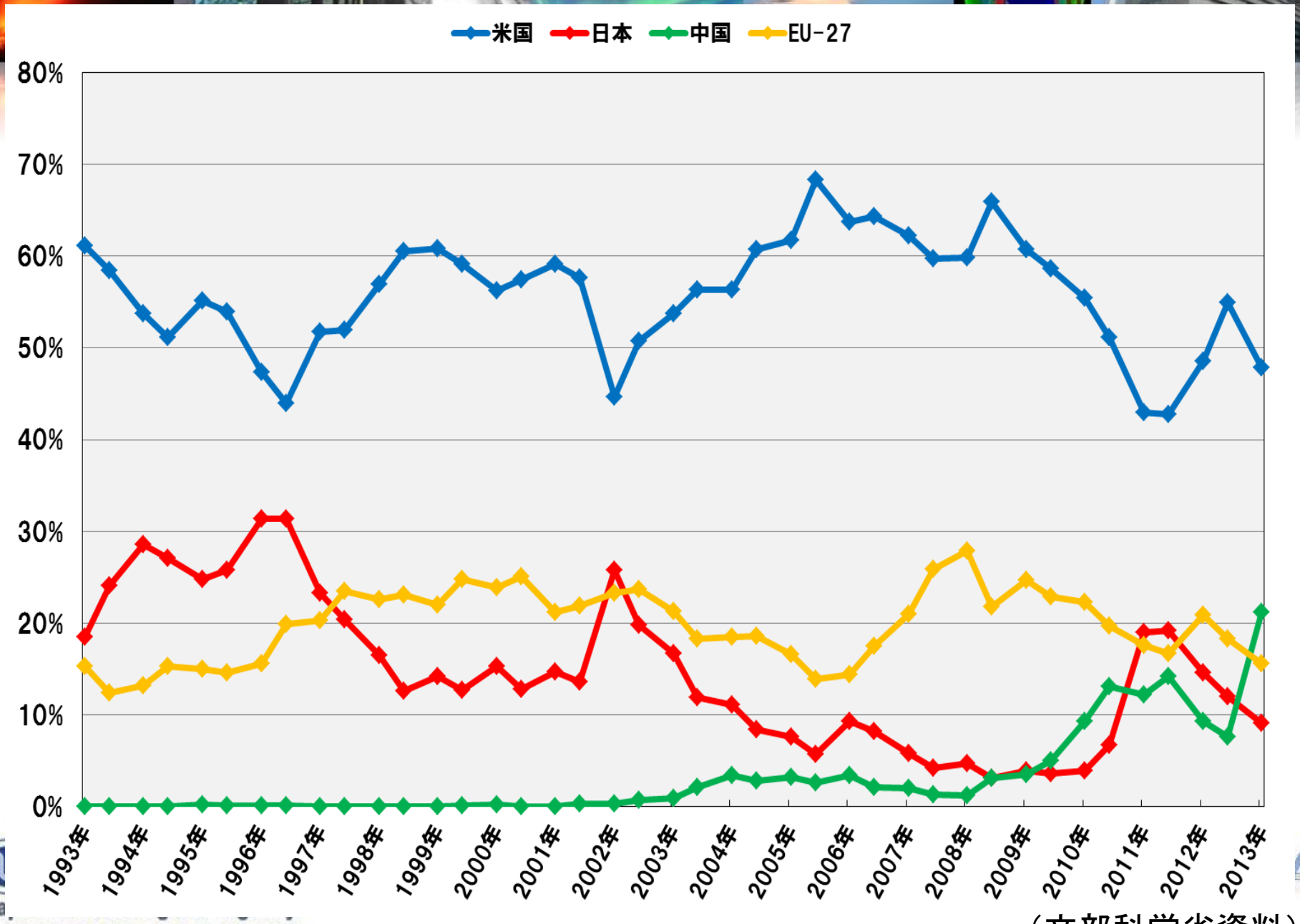
（文部科学省資料）

TOP500の各国1位の推移



（文部科学省資料）

TOP500　国別性能割合推移



（文部科学省資料）

スーパーコンピュータ「京（けい）」の概要

・2011年6月と11月の二期連続で世界スパコン性能ランキング(TOP500)において1位を獲得

・「京」の利用研究が2年連続でゴードン・ベル賞（コンピュータシミュレーション分野での最高の賞）を受賞

○概要

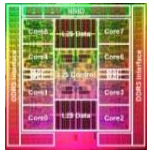
- ◆**平成23年11月にLINPACK性能※1 10ペタフロップス※2達成。**
- ◆平成24年6月システム完成済（兵庫県神戸市の理化学研究所に設置）
- ◆平成24年9月28日に共用開始

※1 スーパーコンピュータの性能を測るための世界的な指標（ベンチマークプログラム，

※2 10ペタフロップス：一秒間に1京回（=10,000兆回＝10^{16}回）の足し算，掛け算が可能な性能

○プロジェクト経費 約1,110億円（H18~H24）

ラック


CPU（富士通製）
8万個以上を使用

○特長

- ◆全CPUフル稼働時の連続実行時間は29時間以上で世界最高水準の**信頼性**
- ◆世界トップ10の**実行効率**(理論性能に対する実際の性能の比率）平均が78%のところ、「京」は93%
- ◆アプリケーションプログラムの**実行性能や使いやすさ**に関して高い性能
- ◆**水冷システム**の導入により消費電力の削減や故障率の低減に寄与
- ◆**六次元メッシュ/トーラス**結合の採用による高い利便性・耐故障性・運用性
- ◆**共用法**に基づき、登録機関（高度情報科学技術研究機構）と理化学研究所が連携し、「京」を利用する体制を構築。

Japan

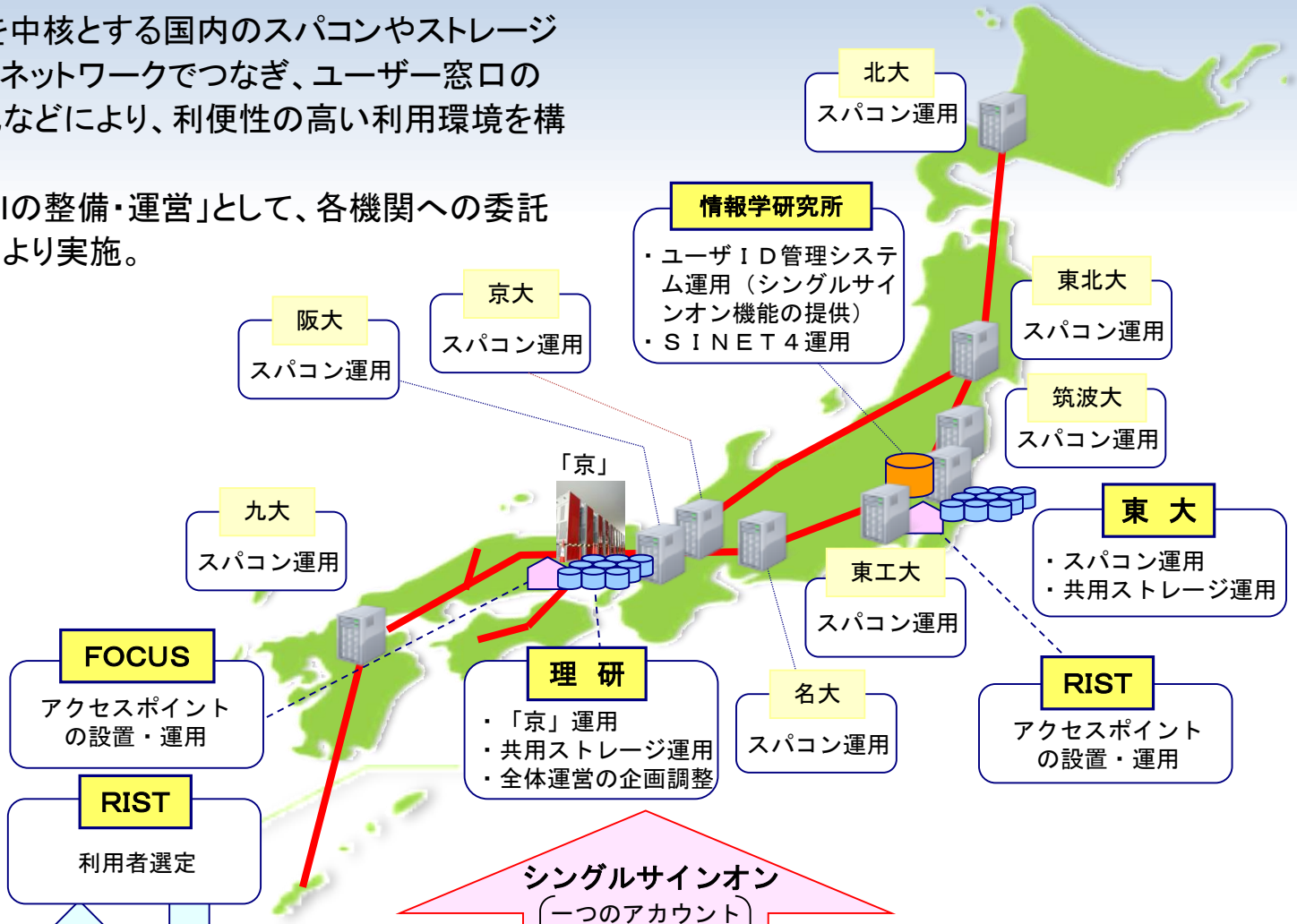
K computer

（文部科学省資料）

HPCIの枠組み

○「京」を中核とする国内のスパコンやストレージを高速ネットワークでつなぎ、ユーザー窓口の一元化などにより、利便性の高い利用環境を構築。

○「HPCIの整備・運営」として、各機関への委託事業により実施。



（文部科学省資料）

9大学情報基盤センターの計算リソースの概要

- 大型計算機を運用管理するとともにその整備を図る
- 学術研究等の共同利用に供する
- 計算機の高度利用に関する研究および開発を行う

平成25年4月現在総理論演算性能　6,509Tflops

北海道大学：
SR16000/M1 (172.6Tflops, 22TB)

大阪大学：
SX-9 (16.4Tflops, 10TB)
SX-8R (5.3Tflops,3.3TB)
Express5800/120Rg-1 (6.1Tflops, 2T
Express5800/53Xh (16.6Tflops, 2.6TB)

東北大学：
SX-9 (26.2Tflops, 16TB)
SX-9 (3.3Tflops, 2TB)
Express5800(1.7Tflops, 3TB)

京都大学：
Cray XE6 (300.8Tflops, 60TB)
APPRO GreenBlade8000 (242.5Tflops, 38TB)
APPRO 2548X (10.6Tflops, 24TB)

筑波大学：
T2K-Tsukuba (95.4Tflops,21TB)
ﾌﾛﾝﾃｨｱ計算機ｼｽﾃﾑ (802Tflops,34TB)

九州大学：
PRIMEHPC FX10 (181.6Tflops,24.6TB)
PRIMERGY CX400 S1 (811.9TF,185TB)
SR16000/L2 (25.3Tflops,5.5TB)

東京大学：
T2K (140.1Tflops, 31TB)
SR16000/M1 (54.9Tflops,11TB)
PREMEHPC FX10 (1135.2Tflops, 150TB)

名古屋大学：
FX1 (30.7Tflops,24TB)
HX600 (25.6Tflops, 10TB)
M9000 (3.84Tflops,3TB)

東京工業大学：
TSUBAME2.0
(2400Tflops,99TB)

Japan Meteorological Agency

（文部科学省資料）

HPCI戦略プログラム戦略分野

「京」を中核とするHPCIを最大限活用し、①画期的な成果創出、②高度な計算科学技術環境を使いこなせる人材の創出、③最先端コンピューティング研究教育拠点の形成を目指し、戦略機関を中心に戦略分野の「研究開発」及び「計算科学技術推進体制の構築」を推進する。

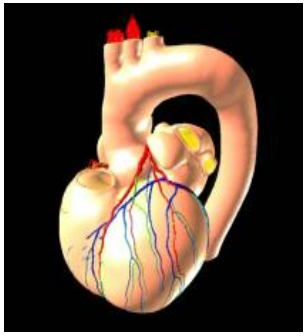
	＜戦略分野＞	＜戦略機関＞
分野1	**予測する生命科学・医療および創薬基盤** ゲノム・タンパク質から細胞・臓器・全身にわたる生命現象を統合的に理解することにより、疾病メカニズムの解明と予測をおこなう。医療や創薬プロセスの高度化への寄与も期待される。	・理化学研究所
分野2	**新物質・エネルギー創成** 物質を原子・電子レベルから総合的に理解することにより、新機能性分子や電子デバイス、更には各種電池やバイオマスなどの新規エネルギーの開発を目指す。	・東大物性研（代表） ・分子研 ・東北大金材研
分野3	**防災・減災に資する地球変動予測** 高精度の気候変動シミュレーションにより地球温暖化に伴う影響予測や集中豪雨の予測を行う。また、地震・津波について、これらが建造物に与える被害をも考慮した予測を行う。	・海洋研究開発機構
分野4	**次世代ものづくり** 先端的要素技術の創成～組み合わせ最適化～丸ごとあるがまま性能評価・寿命予測というプロセス全体を、シミュレーション主導でシームレスに行う、新しいものづくりプロセスの開発を行う。	・東大生産研（代表） ・宇宙航空研究開発機構 ・日本原子力研究開発機構
分野5	**物質と宇宙の起源と構造** 物質の究極的微細構造から星・銀河の誕生と進化の全プロセスの解明まで、極微の素粒子から宇宙全体に至る基礎科学を融合し、物質と宇宙の起源と構造を統合的に理解する。	・筑波大（代表） ・高エネ研 ・国立天文台

※スーパーコンピュータ「京」で、社会的・学術的に大きなブレークスルーが期待できる分野を「戦略分野」（5分野）とする。

（文部科学省資料）

スーパーコンピュータによって期待される成果の例（１）

心疾患のマルチスケール・マルチフィジックスシミュレーション（研究代表者：東京大学・久田俊明）

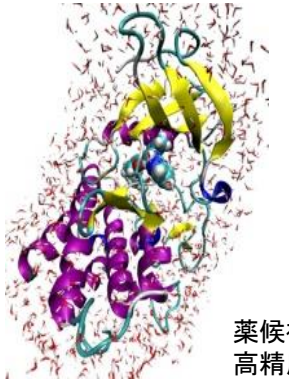


心臓シミュレーション

細胞・組織・臓器を部分ではなく、心臓全体をありのままに再現し、心臓病の治療法の検討や薬の効果の評価に貢献

創薬応用シミュレーション

（研究代表者：東京大学・藤谷秀章）

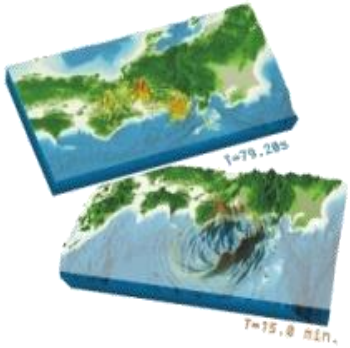


薬候補の高精度結合シミュレーション

新薬の候補物質を絞り込む期間を半減（約2年から約1年）して画期的な新薬の開発に貢献

地震・津波の予測精度の高度化に関する研究

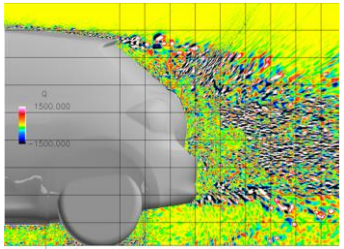
（研究代表者：東京大学・古村孝志、東北大学・今村文彦）



シミュレーションによる地震・津波の被害予測

50m単位（ブロック単位）での予測から地盤沈下や液状化現象等の影響も加味した10m単位（家単位）の詳細な予測を可能とし、都市整備計画への活用による災害に強い街作りやきめ細かな避難計画の策定等に貢献

乱流の直接計算に基づく次世代流体設計システムの研究開発（研究代表者：東京大学・加藤千幸）



車両挙動を解明する全乱流渦のシミュレーション

乱流の直接計算を工業製品の熱流体設計に適用することにより、従来行われていた風洞実験などを完全にシミュレーションで代替し、設計の効率化に貢献

（文部科学省資料）

スーパーコンピュータの課題

- スケーリング
- 消費電力の増大
  - 従来は、電力性能比を犠牲にしても、高速なマシンを求めてきた
    - 「京」・・15MW、一般家庭約3万～4万世帯分
- 信頼性
  - 使用するパーツが増えるとともに、故障率の増大
- プログラミング
  - 分割数が増えると、負荷分散が困難になり、メモリバッファが増大
  - 効率的なライブラリの整備
- IO
  - 容量、消費電力、転送速度、信頼性・・


高速なプログラミング

プログラミング言語

- Fortran90
  - Fortran 95/2003, Co-array Fortran
- C. C++
- Ruby
- Perl, javascript

- GPUコンピューティグ
  - CUDA
  - OpenACC

Fortran の特徴

- 手続型コンパイル言語
  - 計算機言語の保守本流
- 数値計算向け
  - 名前の由来は Formula Translation
- 世界最古の高水準言語
  - 膨大なプログラム資産
  - ベクトル化、並列化などのコンパイラ技術の資産

Fortran 90: 現在

- 規格では FORTRAN から Fortran になった
- 近代的な制御構造の導入（GOTO が不要）
- モジュールの導入（COMMON が不要）
- 配列機能の強化（並列プログラミング）
- 内部副プログラムの導入
- 構造型、ユーザ定義型、ポインタの導入
- 自由形式（キーボードとディスプレイ向き）

配列演算：添字の並べ方

- Fortran の配列 a(i, j) はつぎのようにメモリ上に配置される (i = 1..10, j = 1..5)

	a(1, 1)	a(2, 1)	a(3, 1)	a(4, 1)	a(5, 1)	a(6, 1)	a(7, 1)	a(8, 1)	a(9, 1)	a(10, 1)
……	a(1, 2)	a(2, 2)	a(3, 2)	a(4, 2)	a(5, 2)	a(6, 2)	a(7, 2)	a(8, 2)	a(9, 2)	a(10, 2)
……	a(1, 3)	a(2, 3)	a(3, 3)	a(4, 3)	a(5, 3)	a(6, 3)	a(7, 3)	a(8, 3)	a(9, 3)	a(10, 3)
……	a(1, 4)	a(2, 4)	a(3, 4)	a(4, 4)	a(5, 4)	a(6, 4)	a(7, 4)	a(8, 4)	a(9, 4)	a(10, 4)
……	a(1, 5)	a(2, 5)	a(3, 5)	a(4, 5)	a(5, 5)	a(6, 5)	a(7, 5)	a(8, 5)	a(9, 5)	a(10, 5)

  - 二重の DO ループでは、変数 j の繰り返しを外側に、変数 i の繰り返しを内側にすれば、アクセスがメモリ上で連続になって高速（C 言語の a[j][i] とは i と j の位置が逆）

高速なプログラム

- プログラムは物理法則、数式通りに書くことが基本
- しかし、速く計算する、ということも重要
  - 日々の天気予報はいうまでもないが
  - 研究目的でも迅速性は求められる

- 高速なコンピュータを使うことのほかに、最適化や並列化といった工夫が行われる

コンピュータの性能を決める要素

- CPU の演算速度
- メモリバンド幅
- ノード間（プロセス間）の通信速度

- 近年では CPU の演算速度の向上やメモリ容量の増大に比べてメモリアクセスや通信の速度の向上が遅れている
  - 一回の通信あたりの演算量が多いほうが有利

最適化

- アルゴリズムやデータ構造、ソースコードの書き方を工夫して高速化すること
- アルゴリズムやデータ構造を工夫してメモリアクセスを工夫すること

- 計算時間を多く消費する「ホットスポット」を見つけることがスタート
  - 実行時間の計測、プロファイラの利用

最適化実例
計算順序の入れ替え

- 例：3次式

```
a * x ** 3 + b * x ** 2 + c + x + d
((a * x + b) * x + c) * x + d
```

- 数学的には同じ
- 四則演算回数は上の式は10回、下の式は6回、よって下の式の方が高速

最適化レベルとおもな項目（その1）

- レベル 0, 3, 4
  - べき乗の乗算化

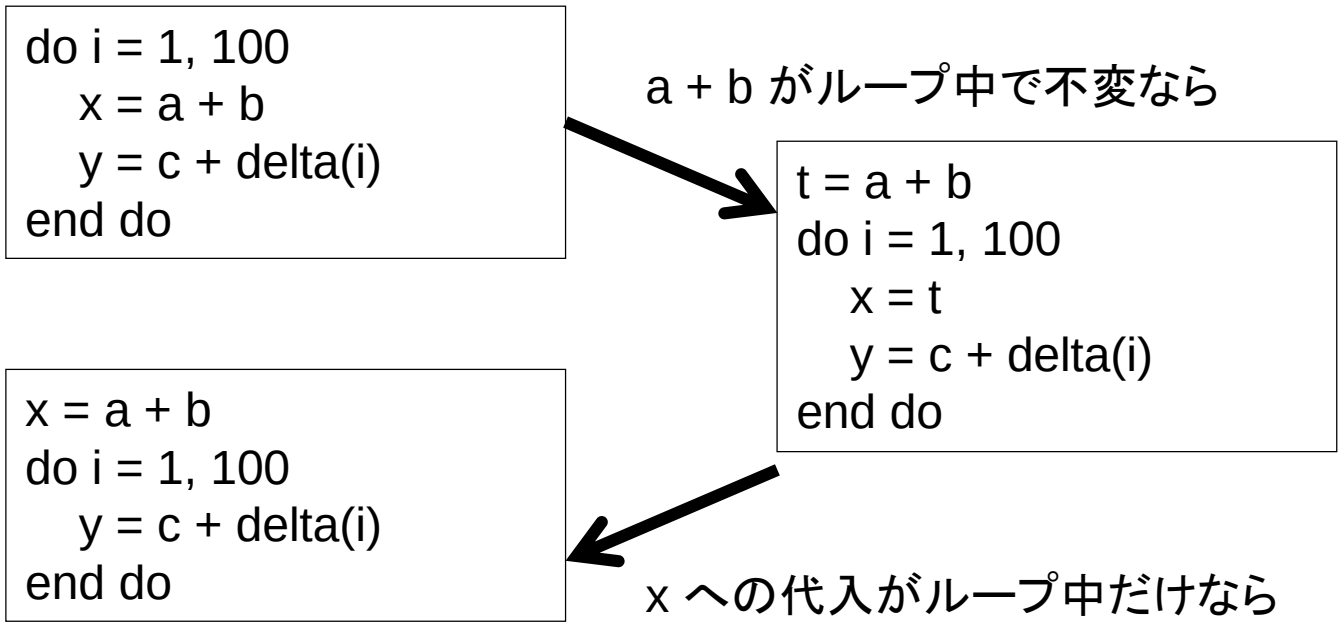
```
a = b ** 3 ⇒ a = b * b * b
```

  - 偶数の乗算にシフト命令を適用
- レベル 3, 4
  - 共通部分式の削除

```
x = a * b
y = a * b * c ⇒ y = x * c
```

最適化レベルとおもな項目（その2）

- レベル 3, 4
  - 不変式のループの外への移動



最適化レベルとおもな項目（その3）

- レベル 4
  - 短いループの展開: 繰り返し回数がコンパイルのときに確定している場合のみ

```
do j = 1, 100
   do i = 1, 3
      a(i, j) = b(i) * c(j)
   end do
end do
```

```
do j = 1, 100
   a(1, j) = b(1) * c(j)
   a(2, j) = b(2) * c(j)
   a(3, j) = b(3) * c(j)
end do
```

最適化が逆効果になることがある

- ループ内の不変式がめったに実行されない IF 文の条件下にある場合

```
do j = 1, 10000
   do i = 1, 10
      if (a(j) > 0.0) then
         a(j) = sin(i * 2.0)
      end if
   end do
end do
```

最適化によって i のループにおける不変式 sin(i * 2.0) の計算が i のループの外に移動された場合、j のすべての繰り返し（10000回）でこの計算が必要になる。
もし IF 文の条件 a(j) > 0.0 が 1度も成り立たなければ、元のコードでsin(i * 2.0) の計算は不要だった。

メモリアクセスの局所性

行列ベクトル積 y = Ax, y_i = sum(A_i,j * x_j)

- オリジナルコード

```
do i = 1, n
    do j = 1, n
        y(i) = y(i) + a(i,j)*x(j)
    end do
end do
```

a(I,j) のアクセスがとびとびになる

改良版

- 連続アクセスに改良

```
do j = 1, n
   do i = 1, n
      y(i) = y(i) + a(i,j)*x(j)
   end do
end do
```

a(I,j) のアクセスが連続になったが、y(i) が毎回呼び出され、書き込まれてしまう

さらに改良

- アンローリングに改良

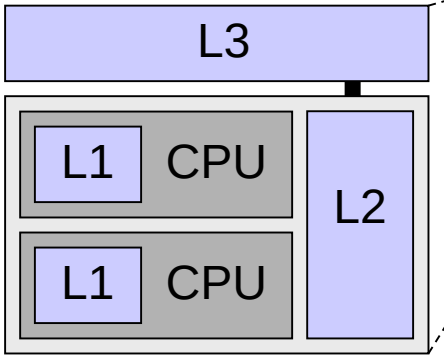
```
do j = 1, n, 2
   do i = 1, n
      y(i) = y(i) + a(i,j)*x(j) + a(i,j+1)*x(j+1)
   end do
end do
```

a(i,j)は連続アクセスで、y(i)の読み書きは半減！j が偶数か奇数かで場合分けが必要

並列化の概要

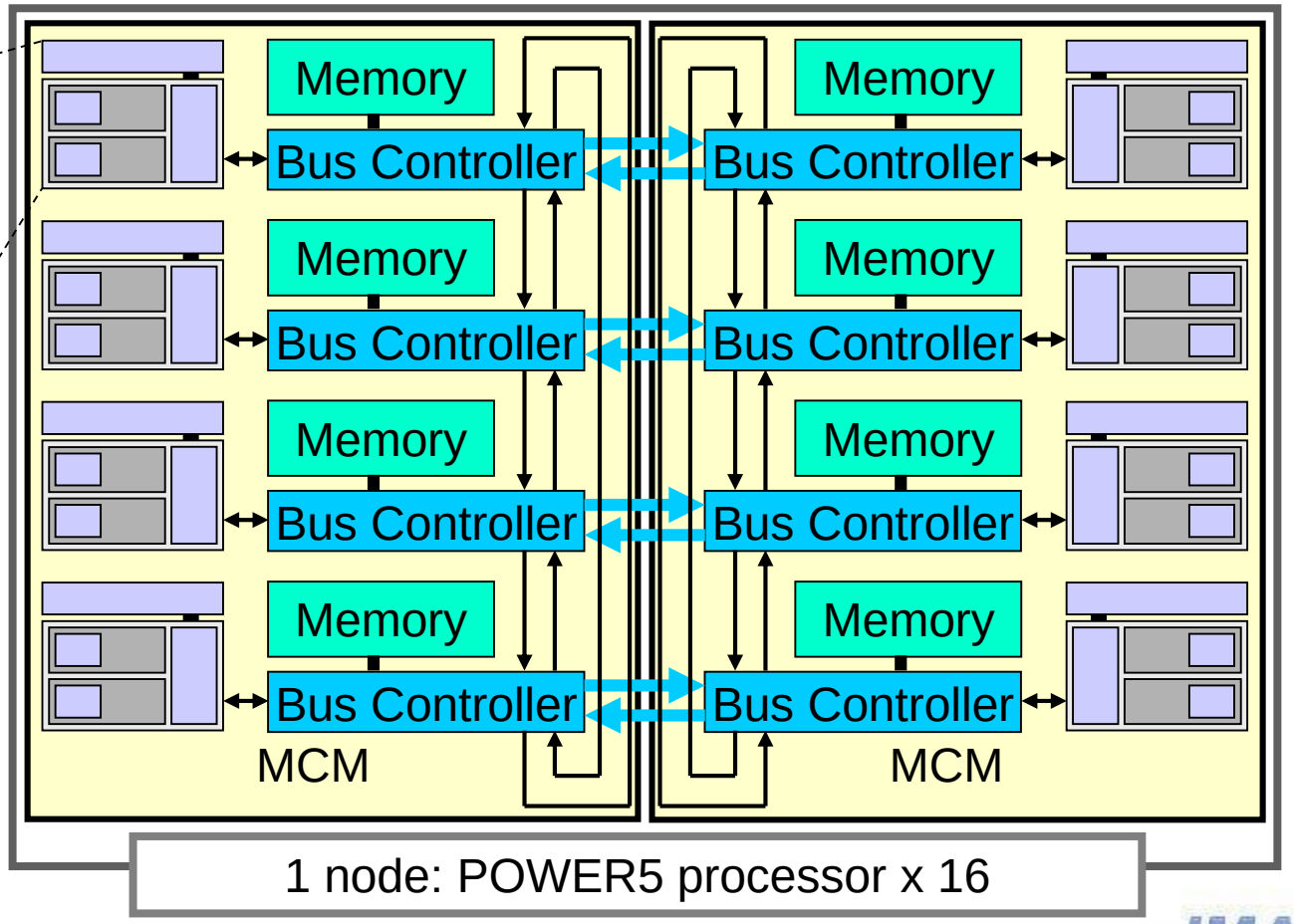
- 計算を複数に分散させる
  - １つあたりの計算量を少なくすることにより処理を高速化
  - 大容量のメモリを利用できるようになる
  - 全体の計算量は変わらないか、オーバーヘッド逆に増える
- ２つの手法（負荷が効率よく分散するように選択される）
  - データパラレル
  - タスクパラレル

並列計算機と多段構造



CPU: 1.9 GHz
/ 2.1 GHz
POWER5

MCM: Multi Chip
Module

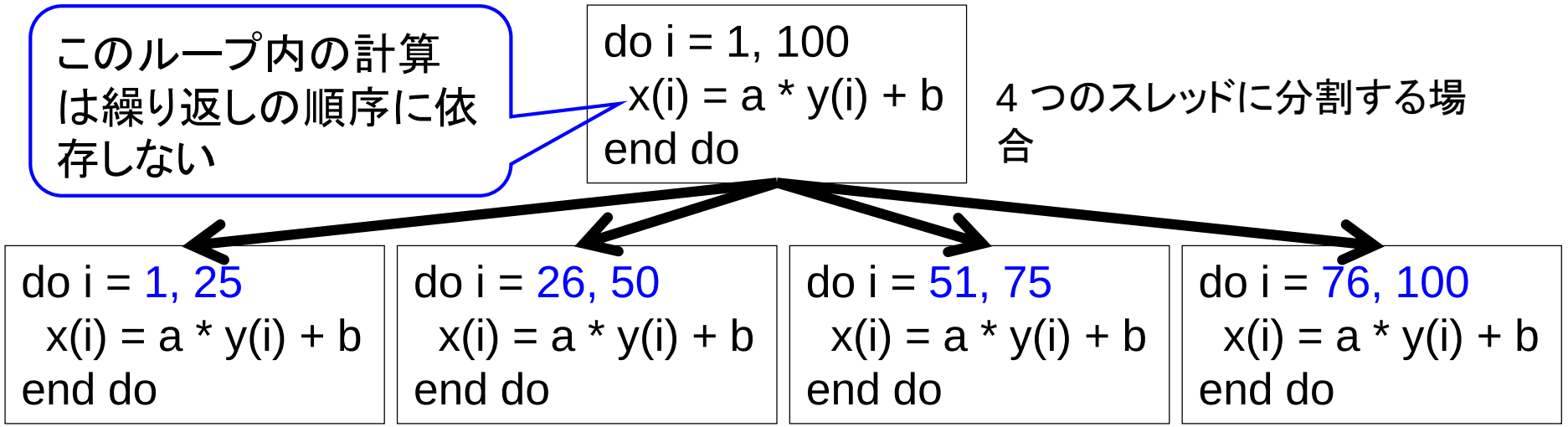


並列化の種類

- 共有メモリ並列化（スレッド並列）
  - コンパイラによる自動並列化（要素並列化）
  - コンパイラに指示行（ディレクティブ）を与える Open MP
- 分散メモリ並列化（プロセス並列）
  - ノード間の通信を明示的に指示する Message Passing Interface (MPI) が標準

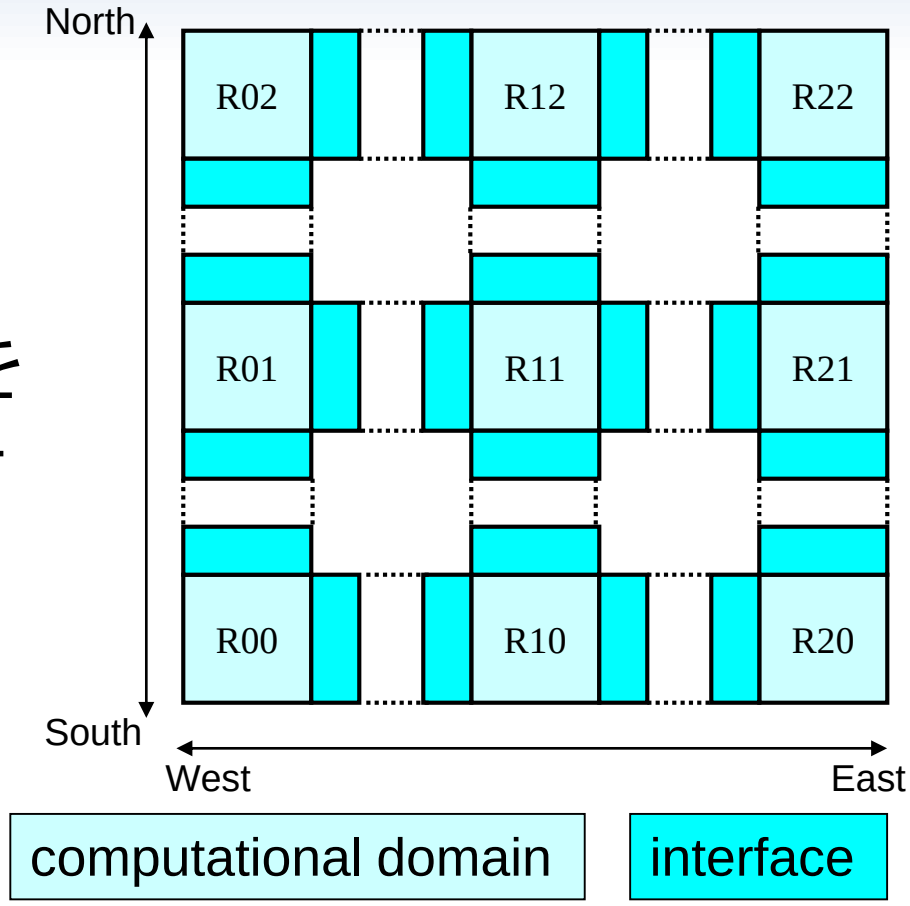
並列化(1)ノード内

- プログラム中の DO ループを複数のスレッドに分割し複数のプロセッサで並列に実行
  - 繰り返しの順序に依存関係がない場合のみ



データパラレル・分散メモリ並列化

- モデルの計算領域を東西方向と南北方向に 2 次元分割
  - 分割した各領域 Rnn を各プロセスで独立に計算
  - 移流など各領域間のやり取りは「のりしろ」(interface) の変数を MPI により通信

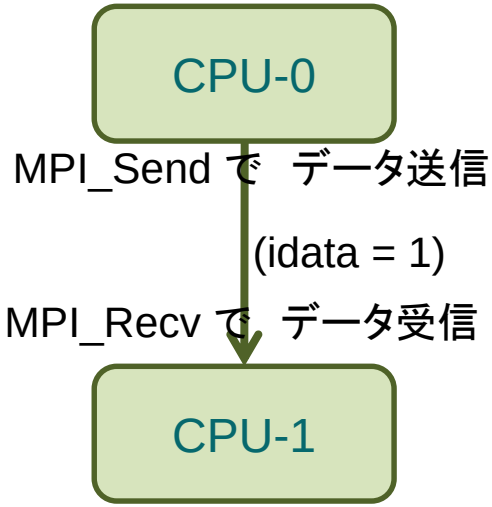


並列化（2）ノード外

```
program mpi_ex   ！超カンタンなプログラム例
  include 'mpif.h'       ! おまじない
  integer :: myrank, idata, ierr
  integer :: status(MPI_STATUS_SIZE)

  call MPI_Init(ierr)                                  ! おまじない
  call MPI_Comm_rank(MPI_COMM_WORLD, myrank, ierr)
  if (myrank == 0) then              ! CPU（ノード）ランクが0なら
    idata = 1
    call MPI_Send(idata, 1, MPI_INTEGER, 1, 1234, &
              & MPI_COMM_WORLD, ierr)  ! データを送信
    write(6,*) myrank, idata
  else if (myrank == 1) then      ! CPU（ノード）ランクが1なら受信
    call MPI_Recv(idata, 1, MPI_INTEGER, 0, 1234, &
                        & MPI_COMM_WORLD, status, ierr)
    write(6,*) myrank, idata
  endif
  call MPI_Finalize(ierr)          ! おまじない
end program
```

コンパイルの例
$ mpif90 mpi_ex.f90 –o mpi_ex



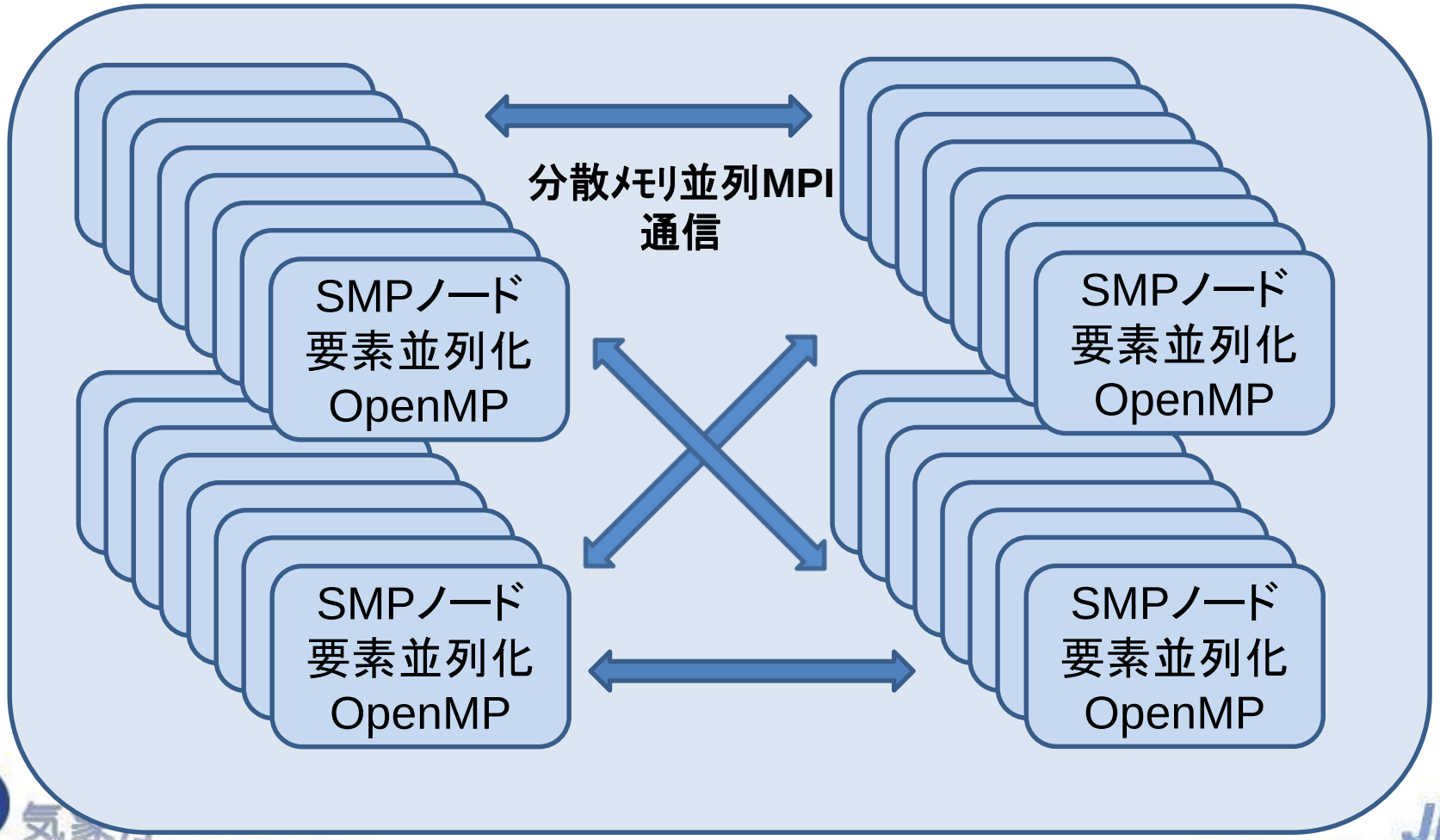
送受信の手続きを
ソースコードに書く!!

実行方法：mpiexec –n 2 ./mpi_ex
実行結果

0	1	（CPU-0の出力）
1	1	（CPU-1の出力）

大きなプログラムを高速で動かすには

要素並列化・OpenMPとMPI通信を上手に組み合わせる技術が必要

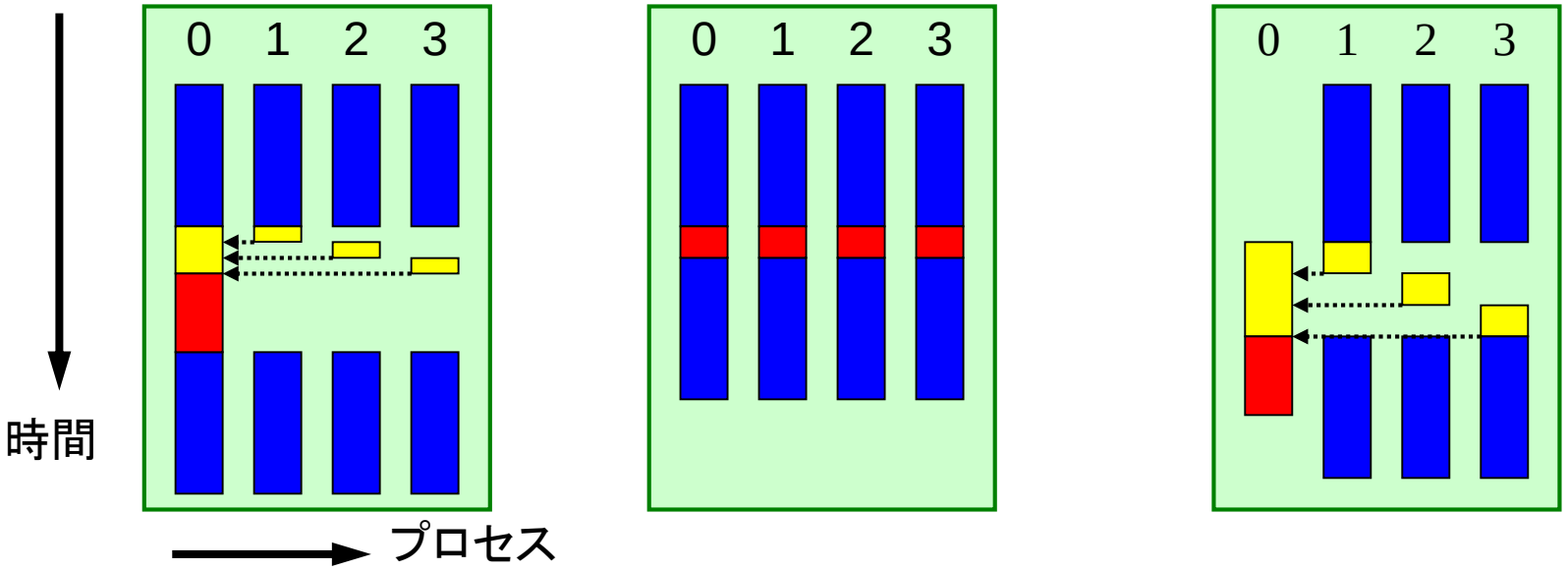


分散メモリ並列化: 出力専用プロセス

communication and output by 0-th proc.

output by each proc.

communication and output by 0-th proc. (for I/O only)



数値計算上の問題

計算機による実数の表現

- 数学的に同じと、数値計算の結果が同じ、は意味が違う
- コンピュータは0と1しか扱えない
- 数学の実数は連続だが、計算機で扱うことができるのは有限桁の浮動小数点数
- ほとんどの計算機が IEEE 754 という規格に定められた単精度と倍精度の表現を採用
  - 単精度（32 ビット）
    - REAL(4)
  - 倍精度（64 ビット）
    - REAL(8) または DOUBLE PRECISION

IEEE 754 が定める浮動小数点数

- 規格は IEEE Std 754-1985 “IEEE Standard for Binary Floating-Point Arithmetic”
  - 単精度（32 ビット）
    - 符号 1 ビット、指数部 8 ビット、仮数部 23 ビット
  - 倍精度（64 ビット）
    - 符号 1 ビット、指数部 11 ビット、仮数部 52 ビット
  - つぎの例外に対する割り込みの有無を指示できる
    - 無効な演算、ゼロによる除算、オーバーフロー、アンダーフロー、不正確

IEEE 754 が定めるゼロ、無限大、非数

- ゼロ
  - 指数部と仮数部のビットがすべて 0
- 無限大
  - 指数部のビットがすべて 1
  - 仮数部のビットがすべて 0
- 非数 (NaN)
  - 指数部のビットがすべて 1
  - 仮数部のビットがすべて 0 ではない

丸め誤差

- 浮動小数点数は実数を有限の長さの 2進数で近似するため、一般に誤差を伴う
  - 丸め誤差 ＝ 浮動小数点値 － 真の値
- 有効数字は有限
  - 10進数の 0.1 を 2進数で表すと循環小数
    0.0001100110011001100 ...
    になる

  ⇒ 10倍しても 1 にならないかも

0.1 の 10.0 倍 == 1.0 ?（4バイト実数）

```
program eqzero4
  implicit none
  real(4) :: val4

  val4 = 0.1
  write(*, *) 'val4 = ', val4
  if (val4 == 0.1) then
    write(*, *) 'val4 == 0.1'
  else
    write(*, *) 'val4 /= 0.1'
  end if
  if (val4 * 10.0 == 1.0) then
    write(*, *) 'val4 * 10.0 == 1.0'
  else
    write(*, *) 'val4 * 10.0 /= 1.0'
  end if
end program eqzero4
```

左のプログラム eqzero4.f90 を SR11000 の日立最適化 FORTRAN でコンパイルして実行する

```
$ f90 -o eqzero4 eqzero4.f90
$ ./eqzero4
 val4 =   0.100000001
 val4 == 0.1
 val4 * 10.0 == 1.0
```

0.1 の 10.0 倍 == 1.0 ?（8バイト実数）

```
program eqzero8
  implicit none
  real(8) :: val8

  val8 = 0.1
  write(*, *) 'val8 = ', val8
  if (val8 == 0.1) then
    write(*, *) 'val8 == 0.1'
  else
    write(*, *) 'val8 /= 0.1'
  end if
  if (val8 * 10.0 == 1.0) then
    write(*, *) 'val8 * 10.0 == 1.0'
  else
    write(*, *) 'val8 * 10.0 /= 1.0'
  end if
end program eqzero8
```

左のプログラム eqzero8.f90 を SR11000 の日立最適化 FORTRAN でコンパイルして実行する

```
$ f90 -o eqzero8 eqzero8.f90
$ ./eqzero8
 val8 =   0.10000000149011619
 val8 == 0.1
 val8 * 10.0 /= 1.0
```

実定数 0.1 は単精度実定数 0.1e0 と同じ

0.1 の 10.0 倍 == 1.0 ?（8バイト実数）

```
program eqzero8d
  implicit none
  real(8) :: val8

  val8 = 0.1d0
  write(*, *) 'val8 = ', val8
  if (val8 == 0.1d0) then
    write(*, *) 'val8 == 0.1'
  else
    write(*, *) 'val8 /= 0.1'
  end if
  if (val8 * 10.0d0 == 1.0d0) then
    write(*, *) 'val8 * 10.0 == 1.0'
  else
    write(*, *) 'val8 * 10.0 /= 1.0'
  end if
end program eqzero8d
```

左のプログラム eqzero8d.f90 を SR11000 の日立最適化 FORTRAN でコンパイルして実行する

```
$ f90 -o eqzero8d eqzero8d.f90
$ ./eqzero8d
 val8 =   0.10000000000000006
 val8 == 0.1
 val8 * 10.0 == 1.0
```

倍精度で扱うためには 0.1d0 とする

情報落ち(積み残し)

- 絶対値が大きく異なる実数の和を計算すると、小さい数が無視される
  - 例

    1000000.0 + 0.0000001

  ⇒ 浮動小数点数: 丸め誤差 ?????? を伴う

    1000000.0??????

  +) 0.0000001??????

    1000000.0??????

  ―――――――――――

  + 0.0000001 は
  丸め誤差に埋没

桁落ち

- 絶対値がほぼ等しい実数の差を計算すると、結果の有効数字が短くなる
  - 例

    1000000.1 − 1000000.0

  ⇒ 浮動小数点数: 丸め誤差 ?????? を伴う

    1000000.1??????

  −) 1000000.0??????

    0.1??????

  ——————————

  結果の有効数字が 1 桁に

桁落ちの実例

- 2次方程式 $ax^2 + bx + c = 0$

  の解は $$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- もし b と $\sqrt{b^2 - 4ac}$ の絶対値がほぼ等しいと、複号 ± のうちどちらかで桁落ちが起こる

桁落ちを避けるために？

- 桁落ちを避けるために、絶対値の大きい解を

$b \geq 0$ ならば　$x_1 = \dfrac{-b - \sqrt{b^2 - 4ac}}{2a}$

$b < 0$ ならば　$x_1 = \dfrac{-b + \sqrt{b^2 - 4ac}}{2a}$

から計算し、もうひとつの解を $x_2 = \dfrac{c}{ax_1}$ によって計算する

気象計算と桁落ち

- 微分方程式、微分で表現される物理量（渦度など）が物理法則にあらわれる気象計算では、桁落ちの問題がよくあらわれる
- 平均場はあらかじめ引いておいて、あとから加える、といった処理が必要

計算順序の入れ替えの影響

- 例：3次式

  a＊x＊＊3＋b＊x＊＊2＋c＋x＋d

  において、べき乗を乗算に置き換え、乗算の回数を減らすため、数学的には等価な

  ((a＊x＋b)＊x＋c)＊x＋d

  に変更すると結果が変わる可能性がある
- 最適化によって計算順序が変わったときには注意（四則演算回数は上の式は10回、下の式は6回、よって下の式の方が高速で、かつ下の式の方が情報おちが少ない）


データ解析、可視化

気象分野の特徴

- 非定常な問題を扱うことが多く、空間3次元・時間1次元の膨大なデータを出力し解析することから、ビッグ・データの問題が常につきまとう
- データ圧縮技術、優れたユーザーインターフェース、人間が理解しやすい応用処理（可視化など）が求められる

データセットのこと

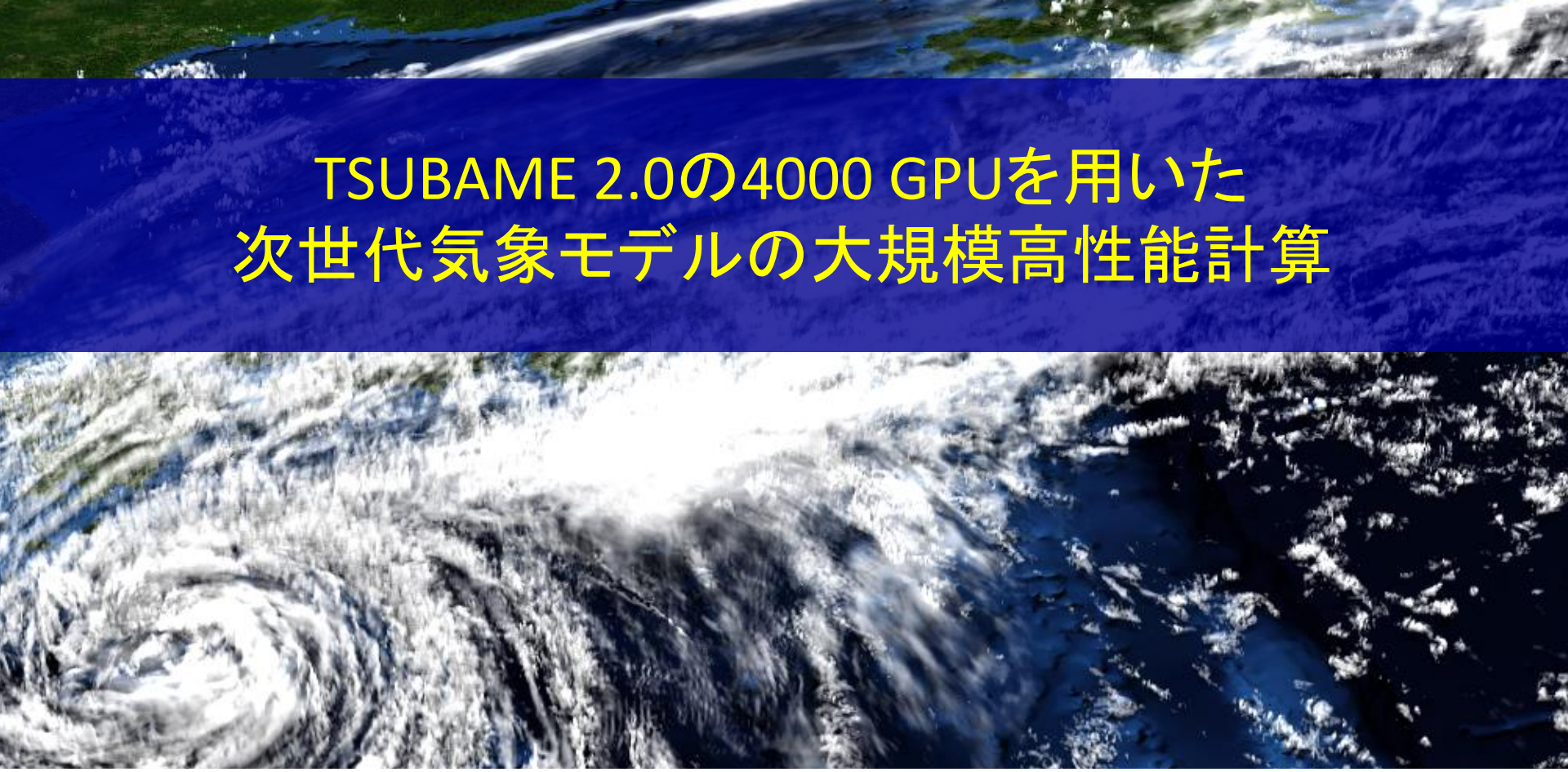
- 効率的な数値予報データ取り扱いのため、様々なデータ形式が提案されている
  - GRIB, GRIB2: 国際交換で標準的に用いられる
  - netcdf: 米国でメジャー
  - nusdas: 気象庁の標準

可視化

- Grads

- 他になにかないのか?
- 個人的にはもう少しなんとかしたい・・


GPUコンピューティング



TSUBAME 2.0の4000 GPUを用いた 次世代気象モデルの大規模高性能計算

下川辺　隆史

東京工業大学

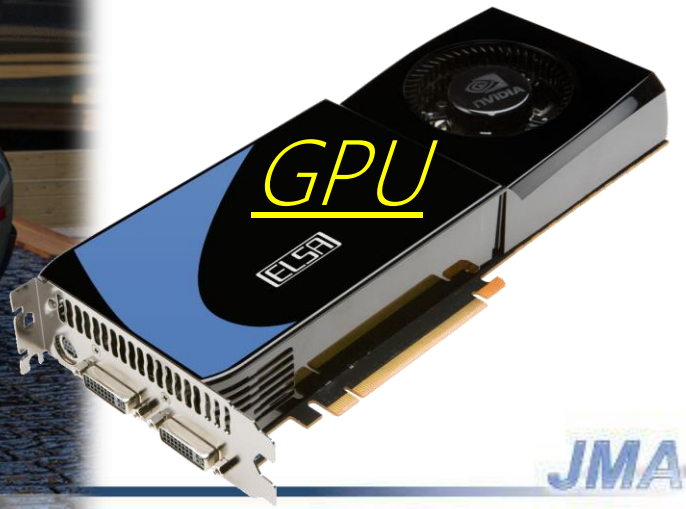
創造エネルギー専攻（学術国際情報センター）

第８回ASE研究会（東京大学情報基盤センター）

Dec. 6, 2010

What's GPU ?

- Graphics Processing Unit
- もともと PC の3D描画専用の装置
- パソコンの部品として量産されてる。
  = 非常に安価



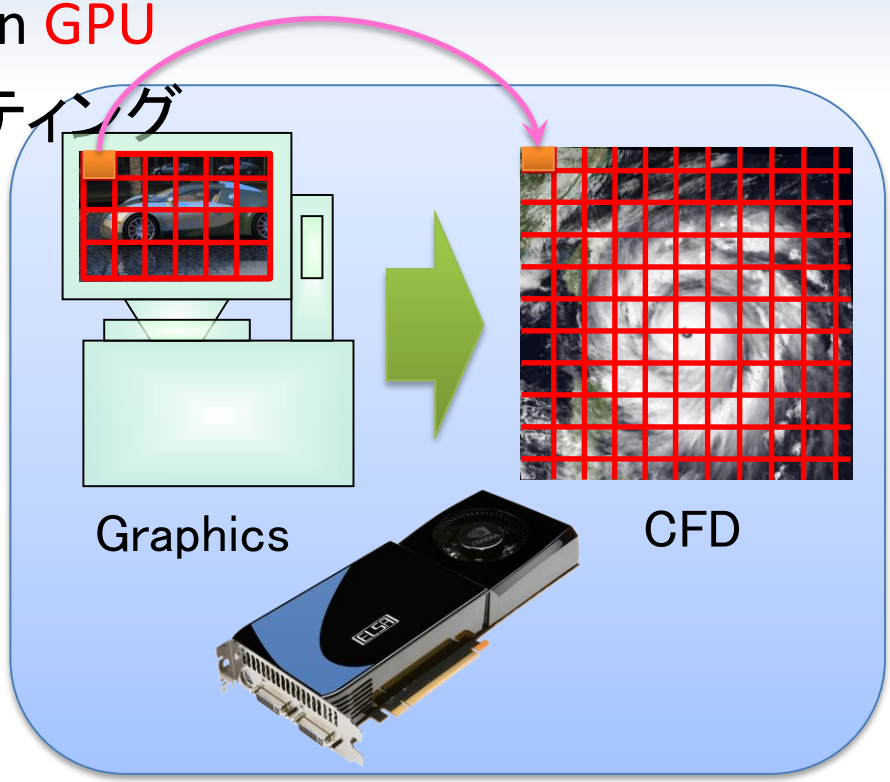
http://www.nvidia.co.jp

GPGPU

GPGPU
http://gpgpu.org/

- General Purpose computation on GPU

  汎用GPU計算、GPUコンピューティング
  - ✓ 数値流体力学（CFD）
  - ✓ N体問題
  - ✓ 高速フーリエ変換（FFT）
  - ✓ …

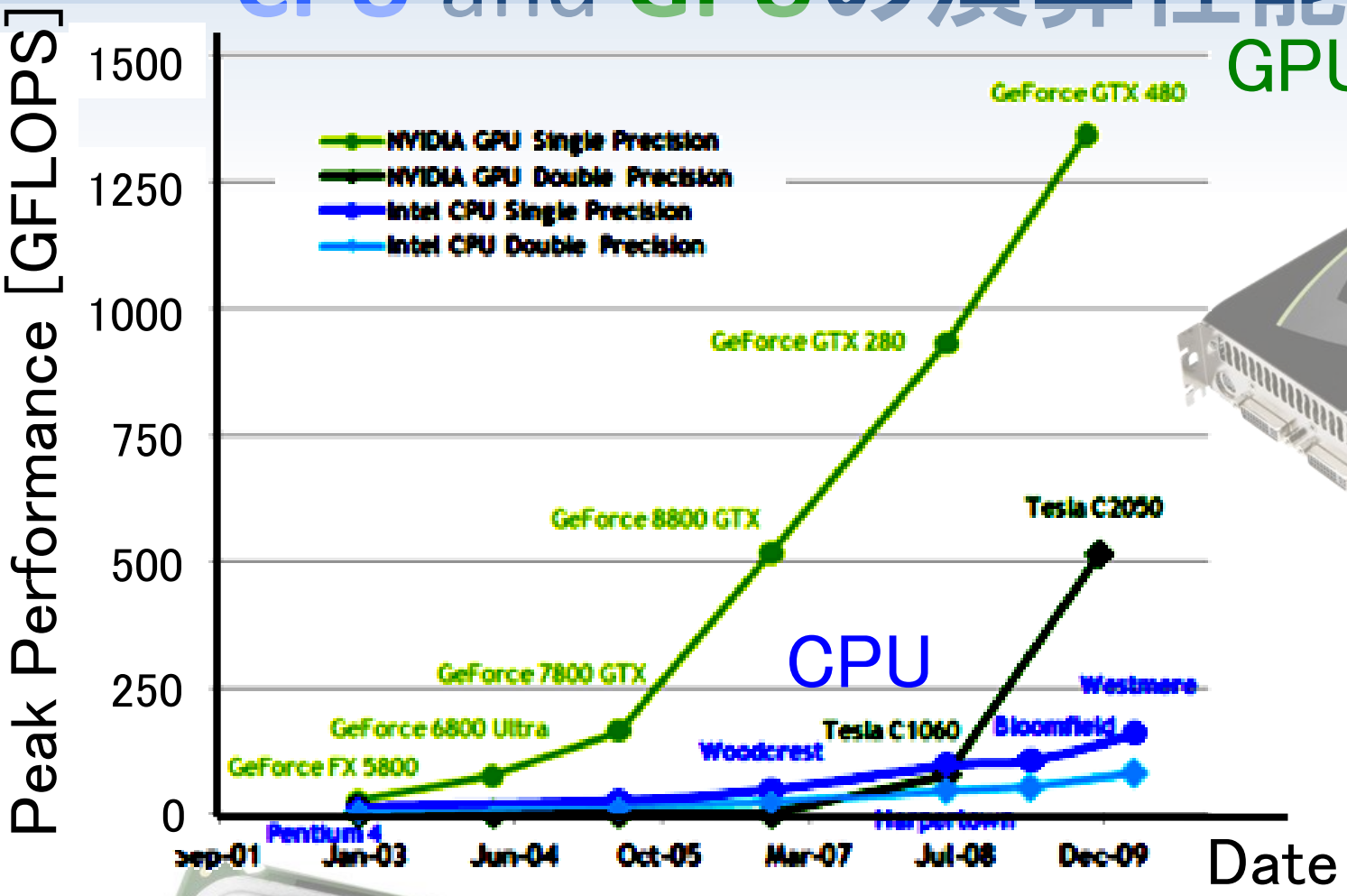


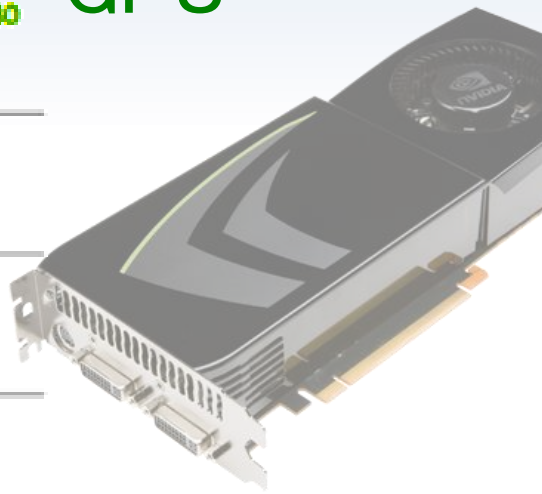
- プログラムはGPUむけの
  開発言語・環境を用いる
  - ✓ CUDA (NVIDIA)
  - ✓ ATI Stream (AMD)
  - ✓ OpenCL (Khronos Group)

GPUによるHigh Performance Computing が現実に。

Japan Meteorological Agency

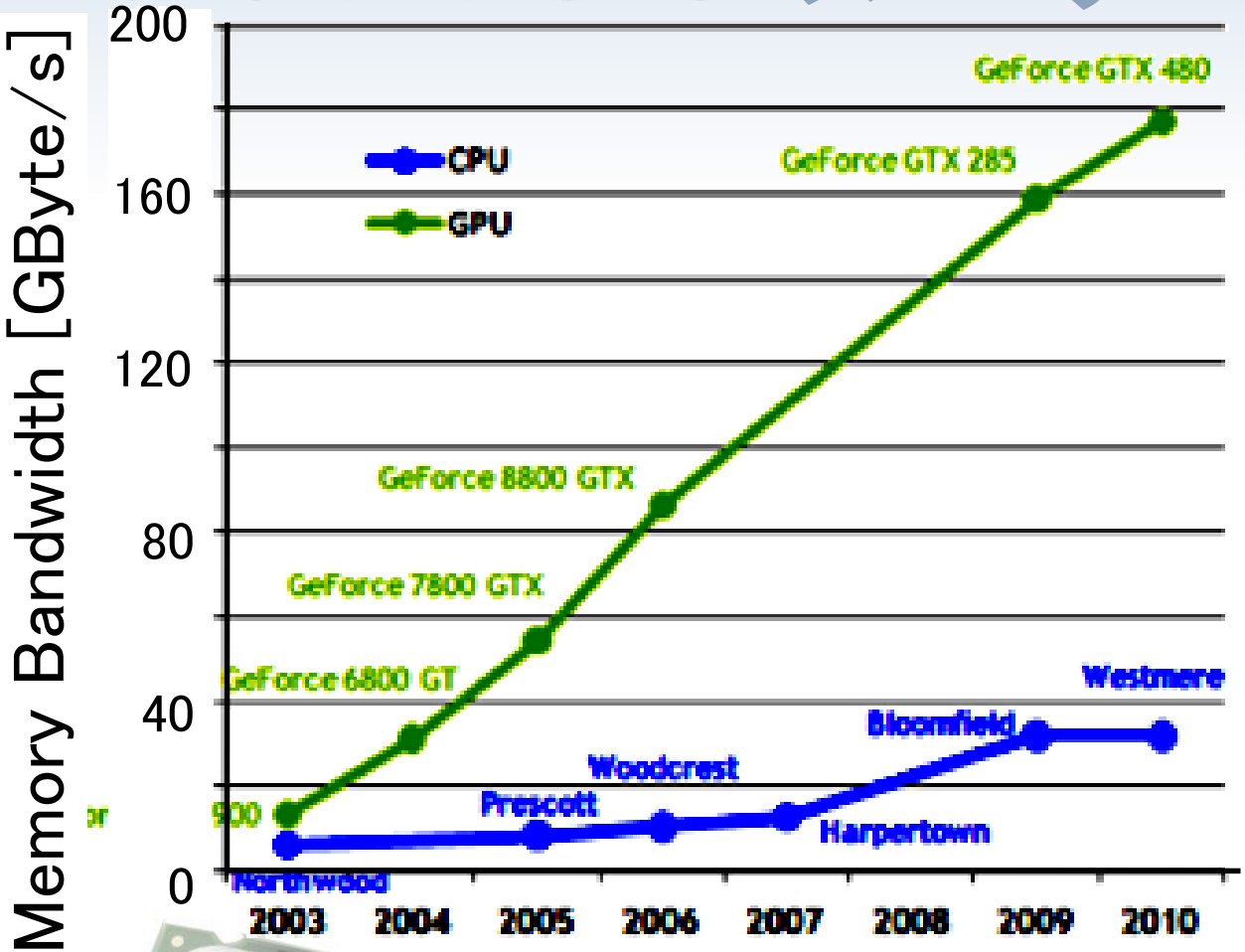
CPU and GPUの演算性能





http://developer.nvidia.com/object/cuda_3_2_downloads.html

CPU and GPUのメモリバンド幅



GPU

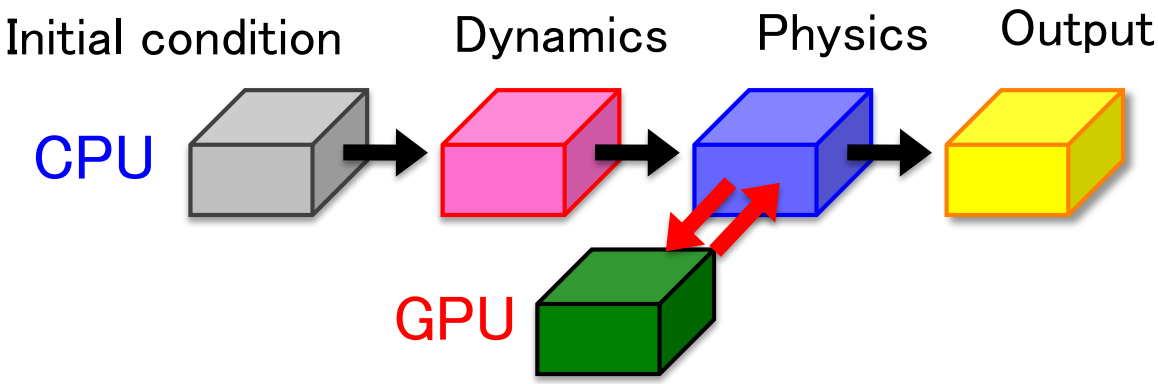
CPU

格子計算のアプリケーションでは多くの場合、
演算性能よりもメモリバンド幅が重要

WRFとASUCAの高速化のアプローチの違い

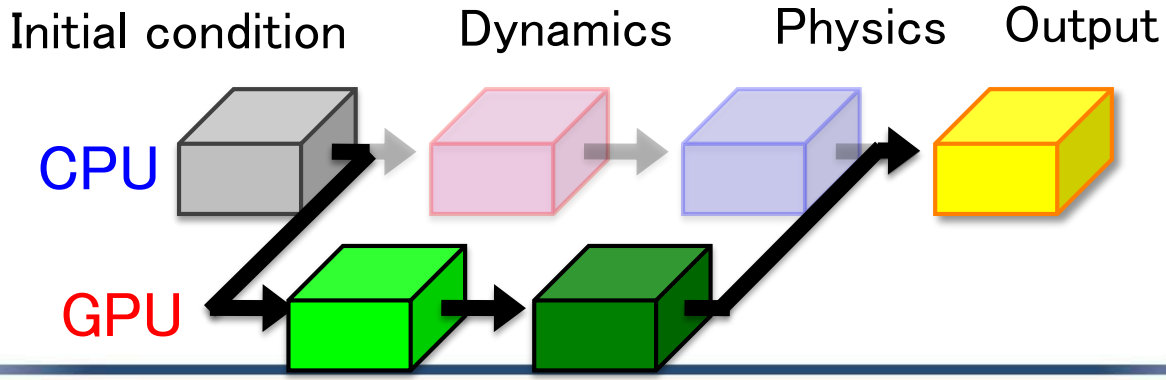
- ✓ WRF GPU Acceleration

Accelerator Approach



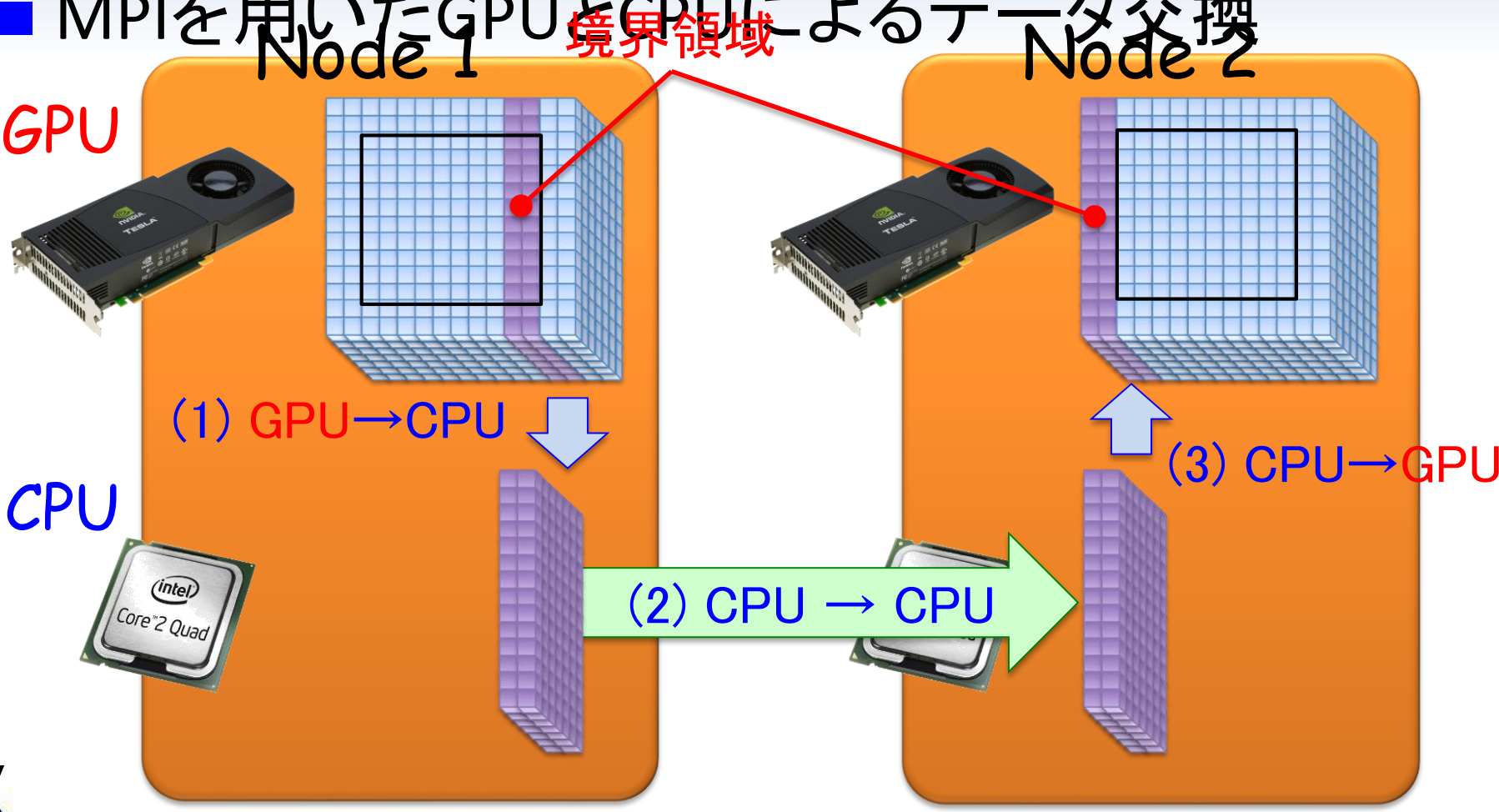
- ✓ ASUCA GPU Computing

Full GPU Approach ➡ 数十倍の高速化の実現



マルチGPU計算：境界領域のデータ交換

- MPIを用いたGPUとCPUによるデータ交換

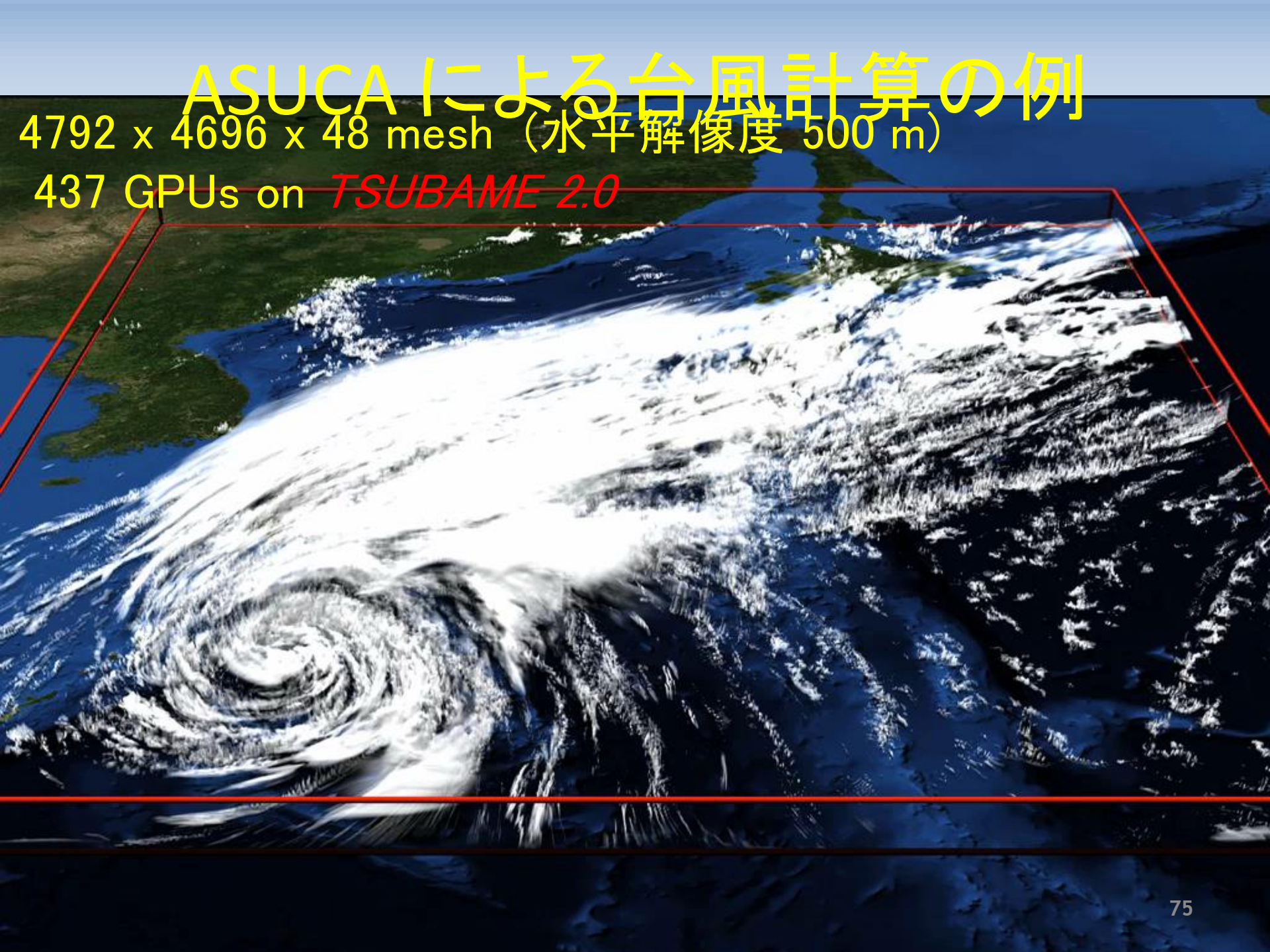


GPUは他のGPU上のメモリへは直接アクセスできない

ASUCAによる台風計算の例

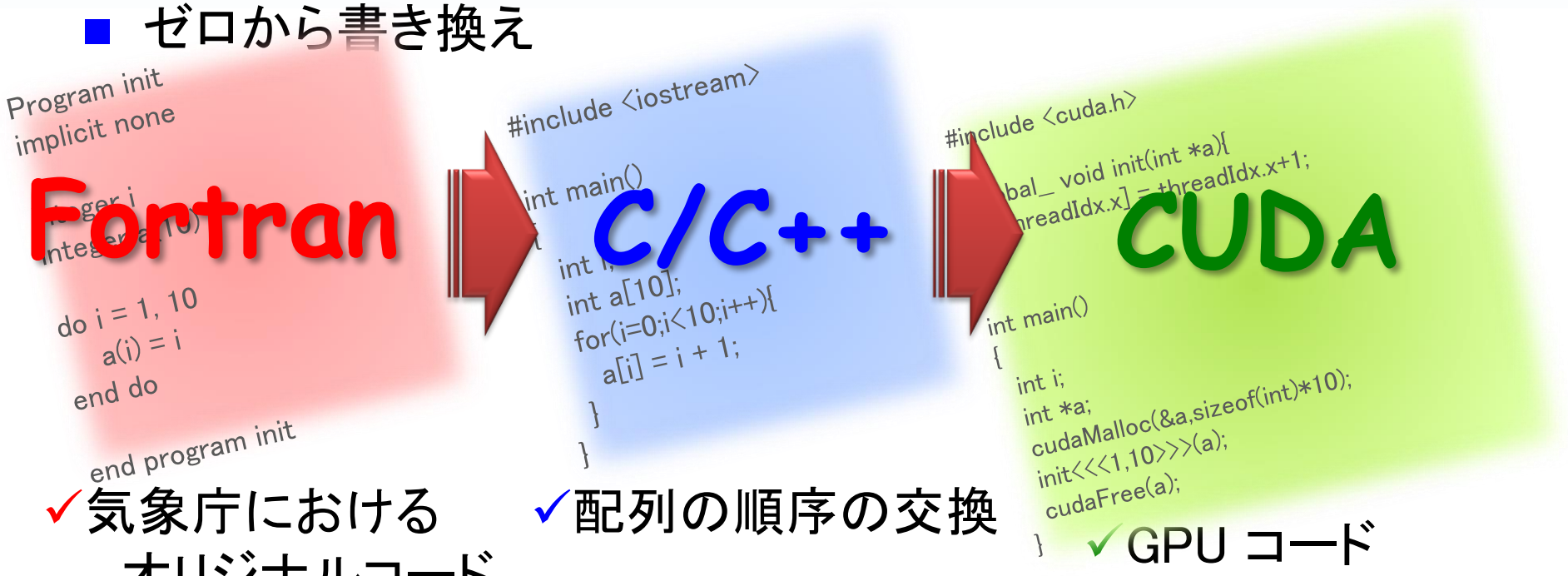
4792 x 4696 x 48 mesh（水平解像度 500 m）

437 GPUs on *TSUBAME 2.0*



ASUCA: Fortran から CUDA へ

- フル GPU アプリケーション
- ゼロから書き換え



✓気象庁における
オリジナルコード

✓配列の順序の交換

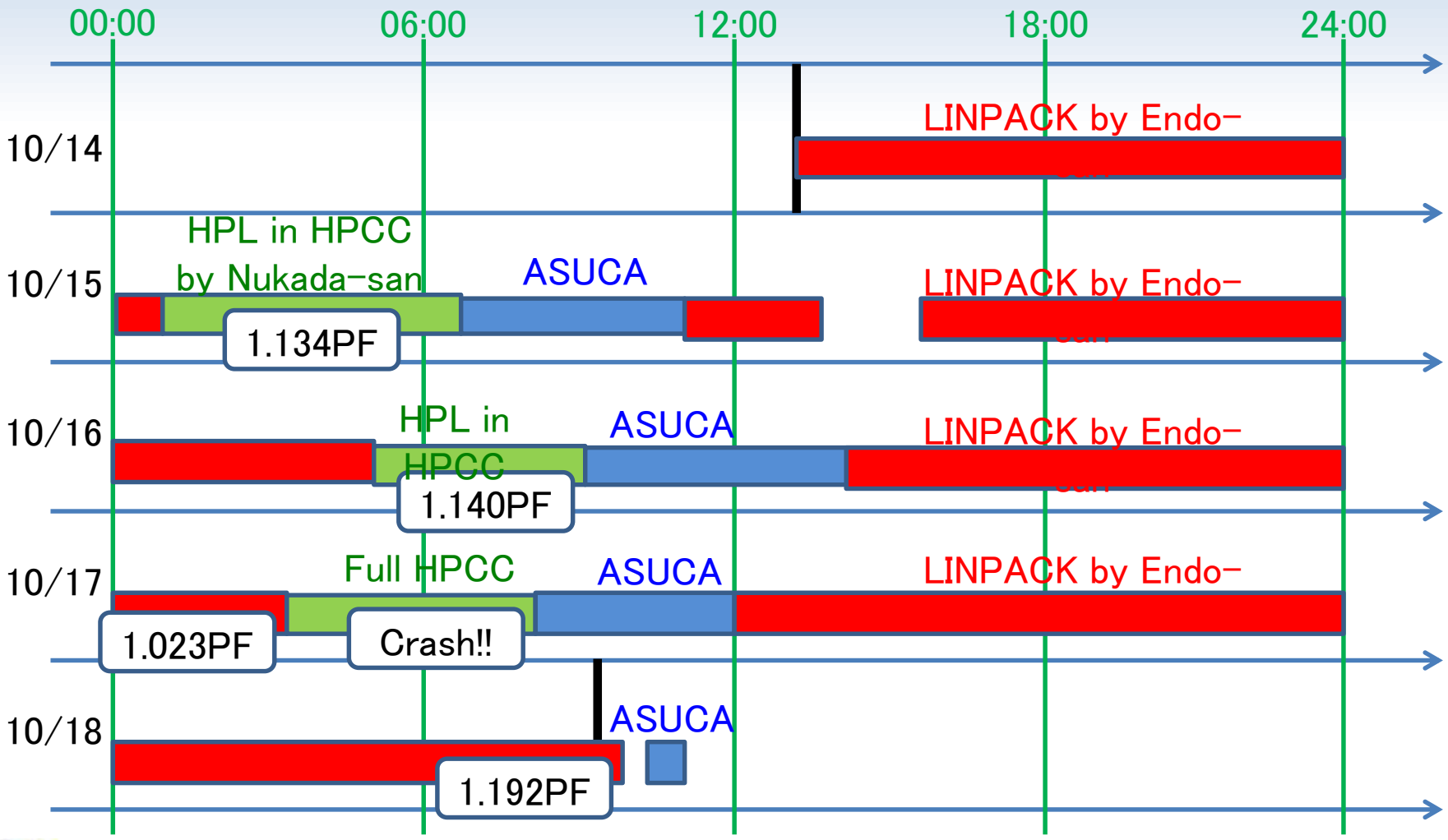
✓GPU コード

[3次元配列の要素順序]

z,x,y (k,i,j)−ordering | *x,z,y* (i,k,j)−ordering | *x,z,y* (i,k,j)−ordering

➡ GPUコードでのメモリアクセスパフォーマンスを向上

TSUBAME 2.0 ベンチマークの日々

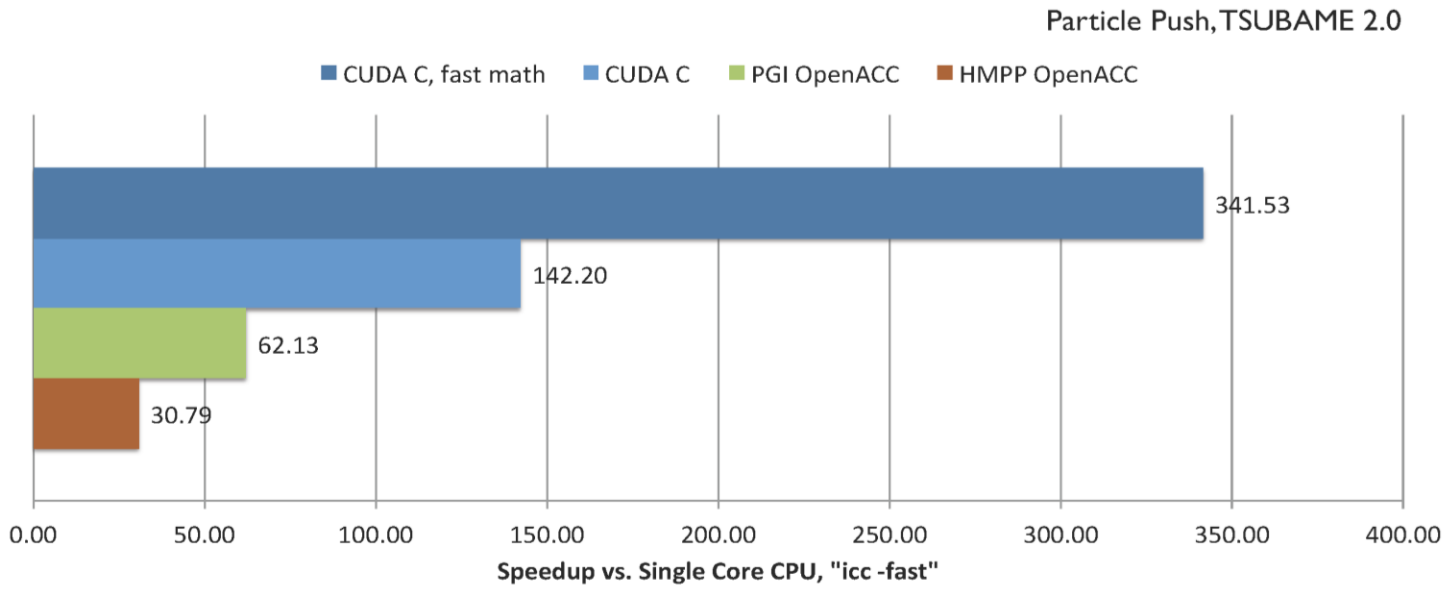


We can use all nodes only for four days.

自動GPU化するOpenACCが登場、しかし遅い

Why not OpenACC?

1. GPU performance for computationally bounded problems



Fortranに指示行を入れて、自動CUDA化、OpenMP化するコンバータを共同開発中