第1回　情報技術研究会

JAXAのFX1システムの紹介と アプリケーションの性能チューニング事例

松尾裕一

宇宙航空研究開発機構
研究開発本部数値解析グループ

2010年5月28日

講演の内容

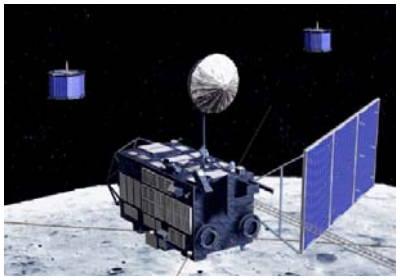
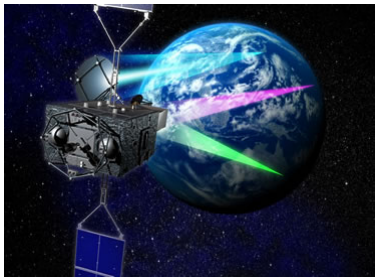
JAXA Supercomputer System

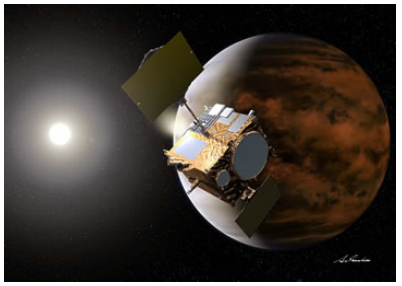
1. JAXA（航空宇宙）におけるHPCアプリ　15分
   - 歴史・経緯
   - 現状と事例紹介
2. JAXAシステム（JSS）の概要　15分
   - 導入経緯，設計思想など
   - システム概要
3. JAXAアプリとそのチューニング　30分
   - プロファイラによる性能情報採取
   - 性能チューニングの例

JAXA-最近の状況

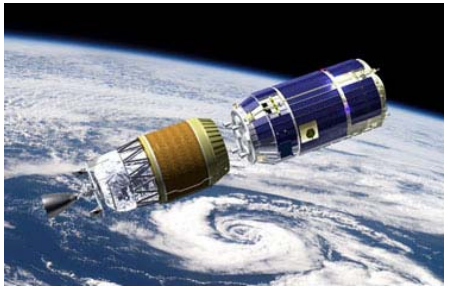
JAXA Supercomputer System

- 衛星打ち上げ…順調
  - 2007.9.14　HIIA13　かぐや（SELENE）　月探査
  - 2008.2.23　HIIA14　きずな（WINDS）　超高速インターネット
  - 2009.1.23　HIIA15　いぶき（GOSAT）　温室効果ガス観測
  - 2009.11.28　HIIA16　情報収集衛星
  - 2010.5.21　HIIA17　あかつき（PLANET-C）　金星探査



- 宇宙ステーション（きぼう），日本人宇宙飛行士
- ISS補給機（HTV）
  - 2009.9.11，HIIBロケット



1. JAXAのHPC・アプリ概要

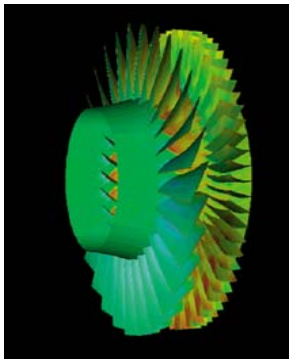
- 主流は航空宇宙のCFD（Computational Fluid Dynamics）
  - 流体現象の解明，空力性能の評価，最適化等の応用
  - JAXAのHPCは航空宇宙アプリとともに歩んできた
- 応用数学の一分野として発展
  - 双曲型保存則，衝撃波捕獲，渦，偏微分方程式
- 現在では工学応用が主体
  - 可能性提示　→　実利用へ
  - 数値風洞以降，JAXA統合で何が変わったか

全盛期（90年代）

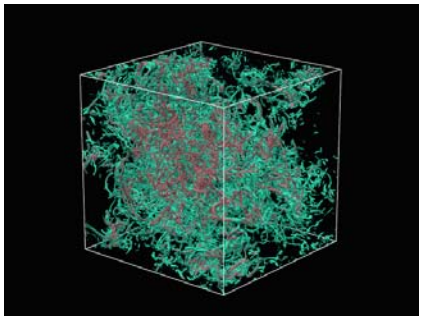
JAXA Supercomputer System

- CFDが最も発展した時期
  - CRAY Y-MP, 2.54GF, 1988-1994（Ames）
  - 数値風洞（NWT）, 280GF, 1993-2002（NAL）
  - ゴードンベル賞，1994-1996

- ベクトルCFDコードの開発・成熟
  - 内容は可能性提示に留まった
    - 実利用はまだまだ
  - 一定の学術的成果



エンジン内部流の非定常解析



512^3の一様等方性乱流DNS

JAXAにおけるアプリの進化

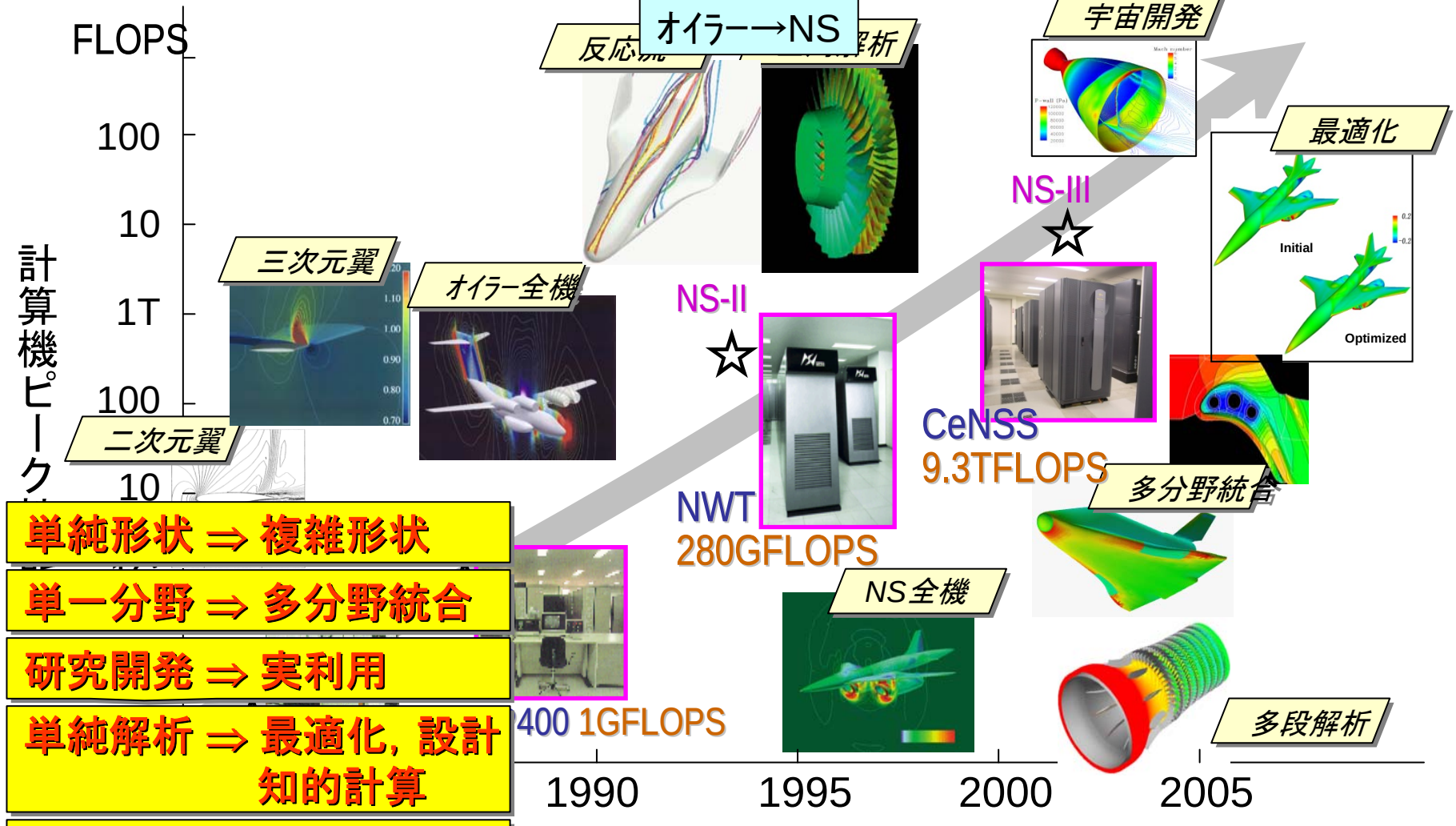
構造格子→非構造格子

AMR

単一領域→複数領域

粒子法

計算機の進歩と相互啓発的に発展

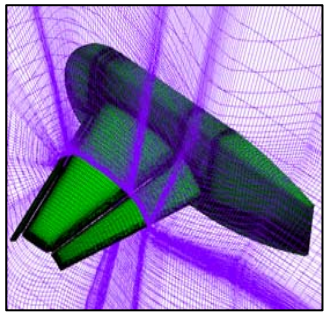


- 単純形状 ⇒ 複雑形状
- 単一分野 ⇒ 多分野統合
- 研究開発 ⇒ 実利用
- 単純解析 ⇒ 最適化，設計知的計算
- 単独利用 ⇒ 遠隔，ｸﾞﾘｯﾄﾞ

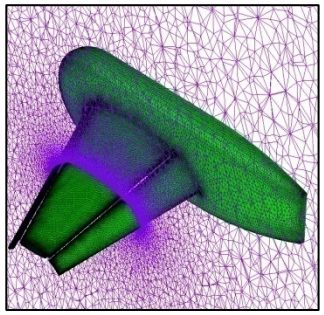
CapacityとCapability

JAXA Supercomputer System

Capacity指向 - 工学系



マルチブロック構造格子

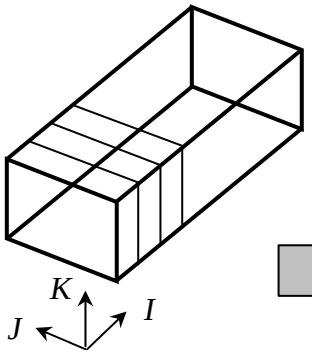


非構造格子

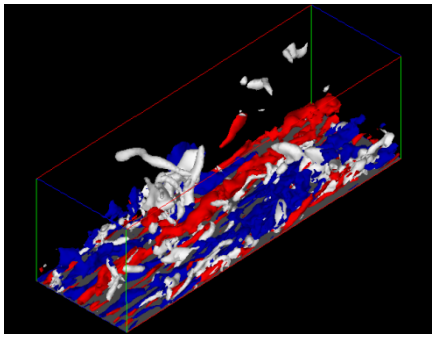
RANS, URANS,
DES, LES
5M～50M点
5時間～数日

- 流体,熱,構造,連成,最適化,パラスタ
- 格子点法中心，市販コード利用
- マルチブロック構造格子，重合格子，非構造格子，直交格子
- 領域分割並列，MPI，ロードインバランス
- 少量（表面のみ）だが複雑な通信
- 並列度小～中（<100）
- ターンアラウンド重視，多数ジョブ（パラスタ）
- 少量のI/Oを多数回
- 大量時系列データ，多数ファイル

Capability指向 - 学術系



ループ並列

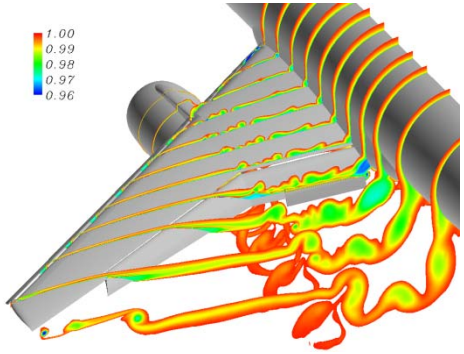


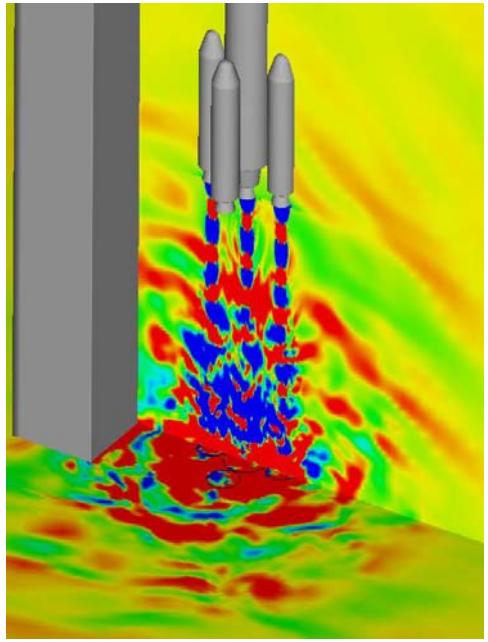
DNS, LES
10M～1G点
1週間～数ヶ月
数100GB ～TB
出力データ

- 流体,プラズマ,宇宙科学,天文
- 格子点法，粒子法，FFT
- 単一領域，構造格子
- ループ並列，XPF，MPI
- 転送は単純だが大量，軸の持替え
- 並列度大（>100）
- 規模（メモリ）重視，とにかく答え
- 頻度少ないが大量のI/O
- 大量出力データ，少数ファイル

JAXAにおけるHPCの現状俯瞰

- 航空機，ロケット，衛星・探査機の設計・開発支援
  - 信頼性向上，開発期間短縮，コスト削減，先進技術の開発
  - 数値シミュレーション技術の活用を重点化
    - 基礎実験/データ，打ち上げ実績： 欧米と大きな差のため
- 学術研究のツール
  - 宇宙科学，空気力学を中心に
- アプローチ
  - 課題解決
    - 現象理解（極限状態）
    - モデル化



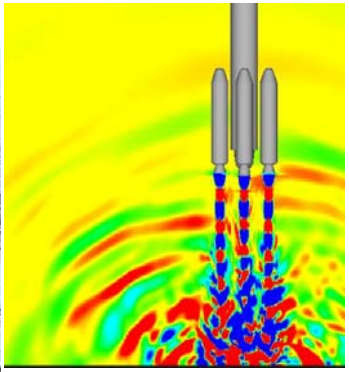


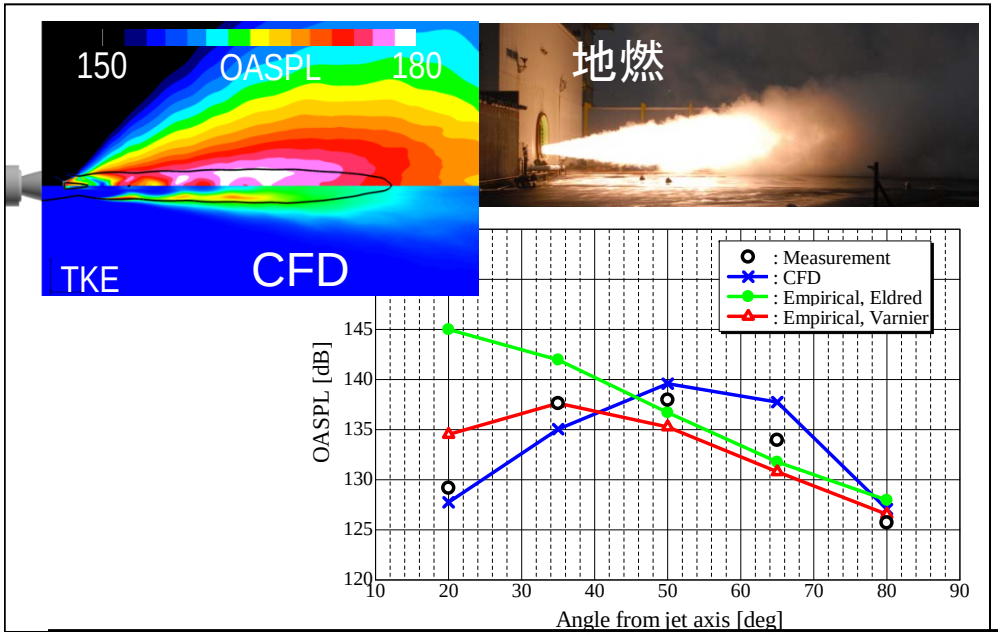
  - 試験の代替，設計プロセスの革新
    - 概念検討，最適化（設計探査）

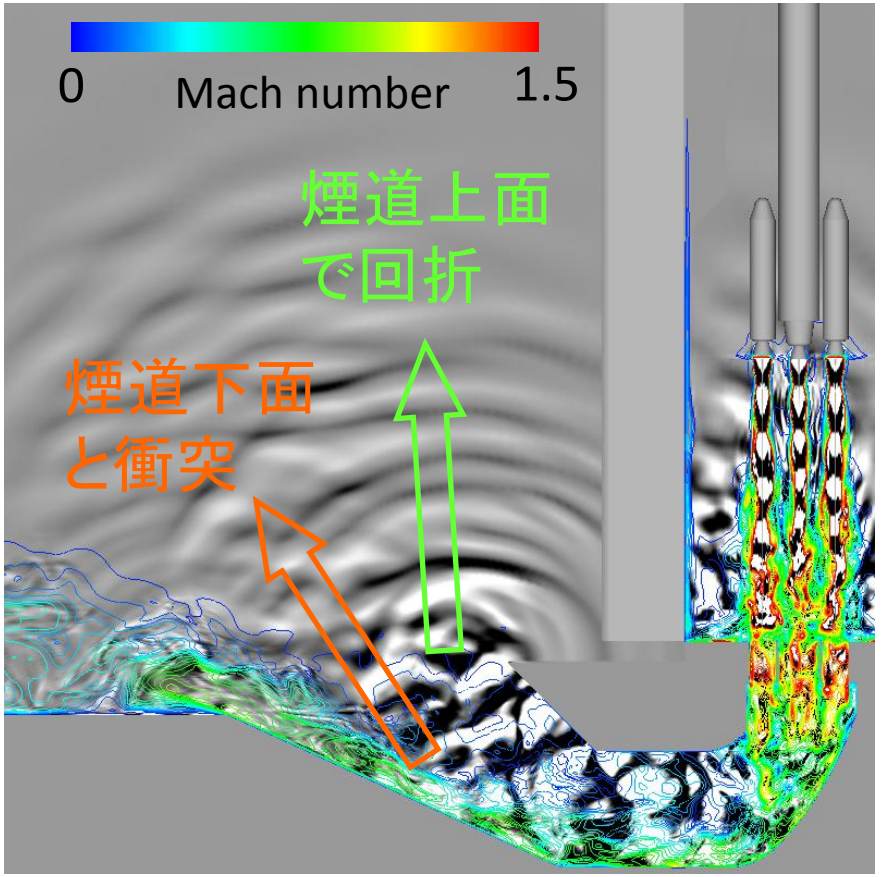
事例1:　音響環境の解析・予測

JAXA Supercomputer System

- ロケット打ち上げ時のプルーム排気による騒音の予測





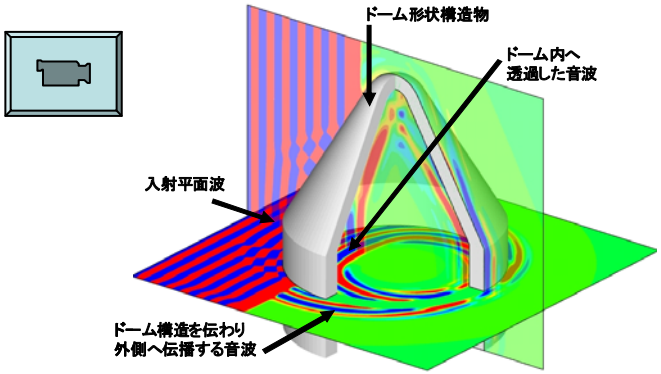


ダウンレンジにおける圧力波の発生/伝播

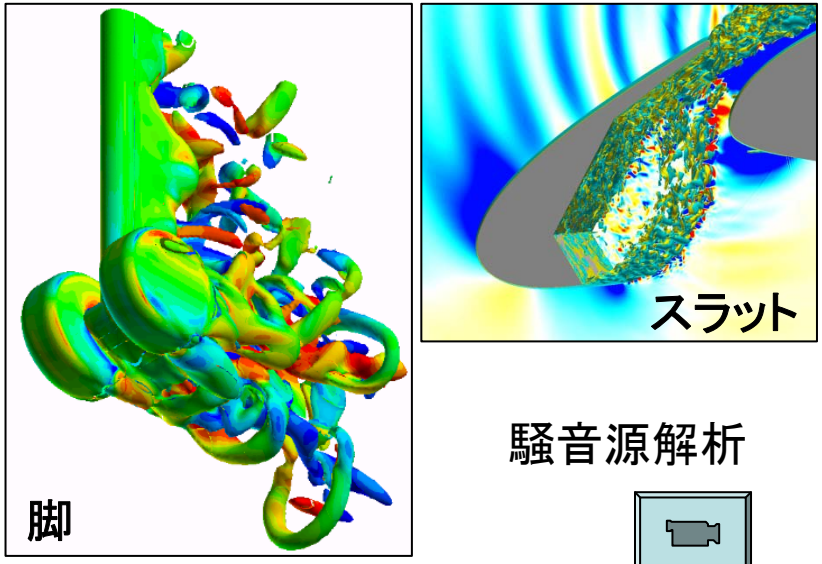
事例1:　音響環境の解析・予測(2)

JAXA Supercomputer System

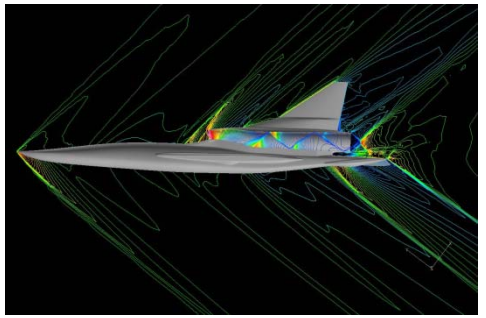
- 宇宙機・航空機の機体内外騒音の予測・低減



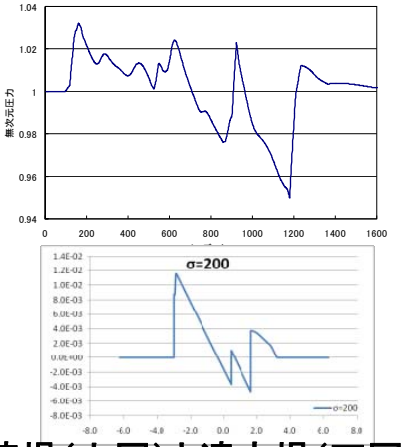
構造物への音の透過や音による構造物の振動の解析



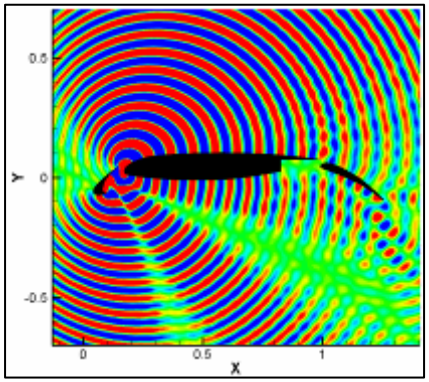
騒音源解析



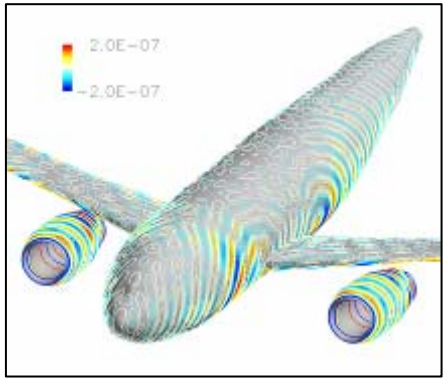
ソニックブームの近傍場/
遠方場解析



近傍場(上図)と遠方場(下図)
の波形



騒音伝播解析



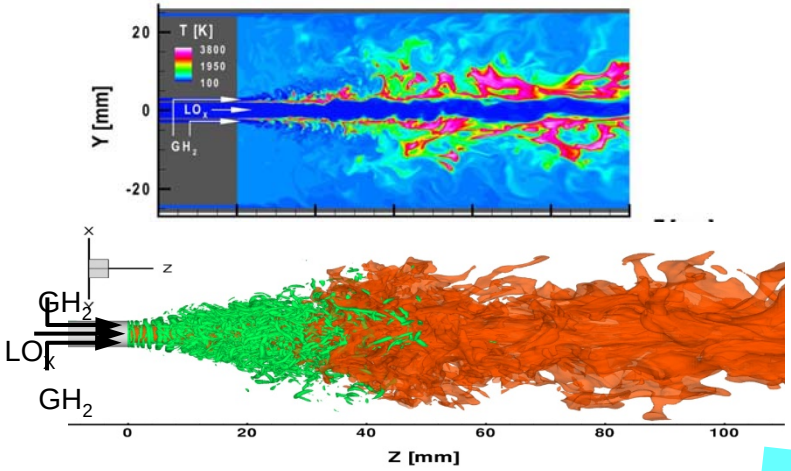
事例2： 燃焼の解明，不安定性の予測

JAXA Supercomputer System

- ロケットエンジン燃焼器等の燃焼不安定や噴霧燃焼の解明

超/亜臨界燃焼
非定常解析

音響／火炎応答解析



超臨界圧LOx/H_2同軸噴射エレメント

乱流燃焼モデリング

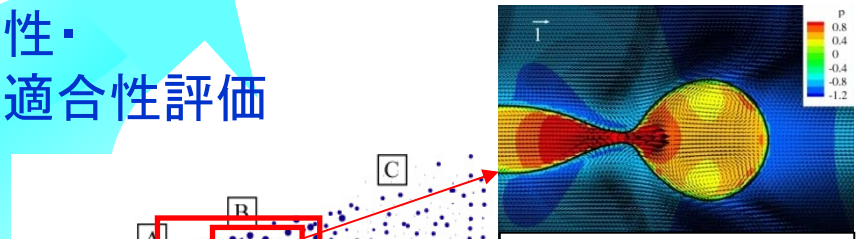
- 温度，発熱，組成
- 乱流効果
- 排出物

予測・定式化

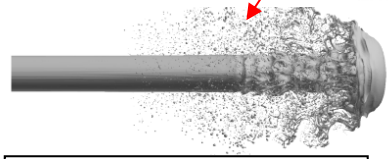
安定性・
環境適合性評価

サブスケール
／実スケール
燃焼器非定常解析

次期中期計画



表面張力波による
液糸からの液滴生成



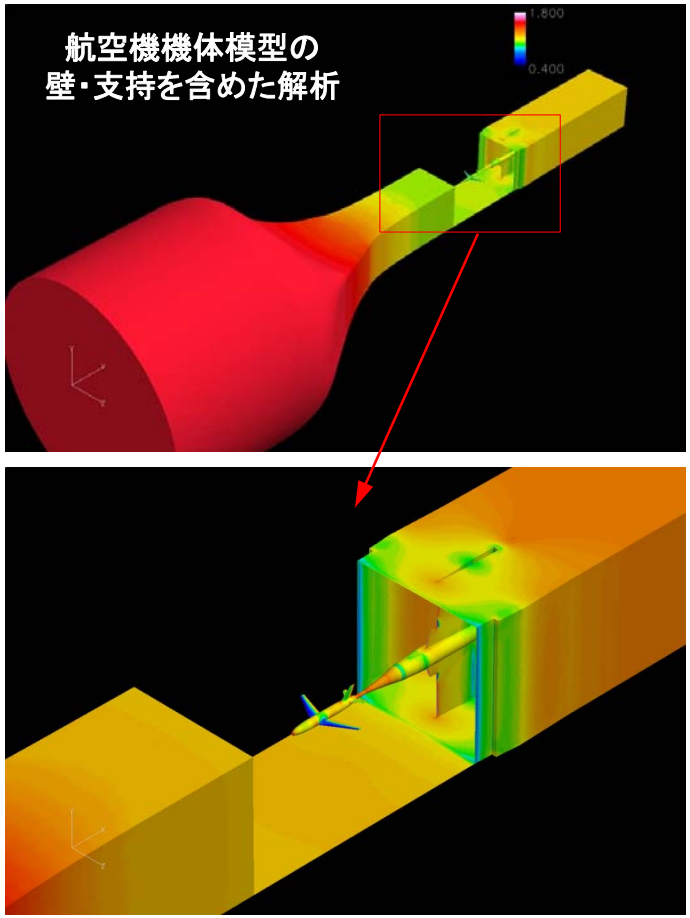
燃料噴流からの液糸生成

液体燃料微粒化過程
解明・モデル化

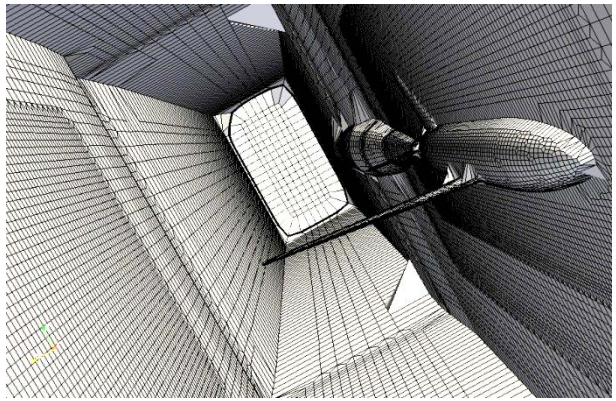
事例3： EFD/CFD融合

- 実験流体力学（EFD）と計算流体力学（CFD）の融合により
  それぞれの弱点を補完，試験効率化 ⇒ ハイブリッド風洞

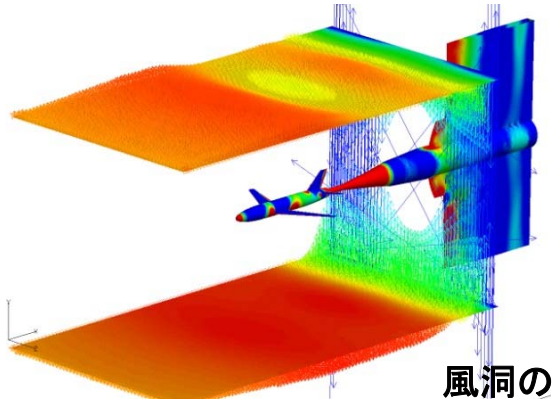
風洞全体（壁・支持を含む）の解析



自動格子生成・高速ソルバー



自動で生成された風洞内の格子



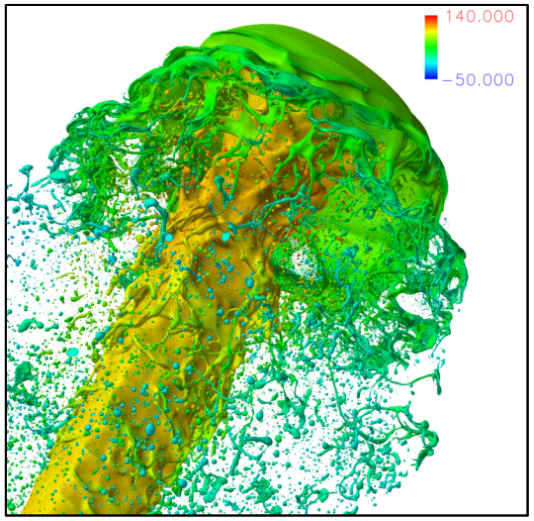
風洞の多孔壁を含む計算結果

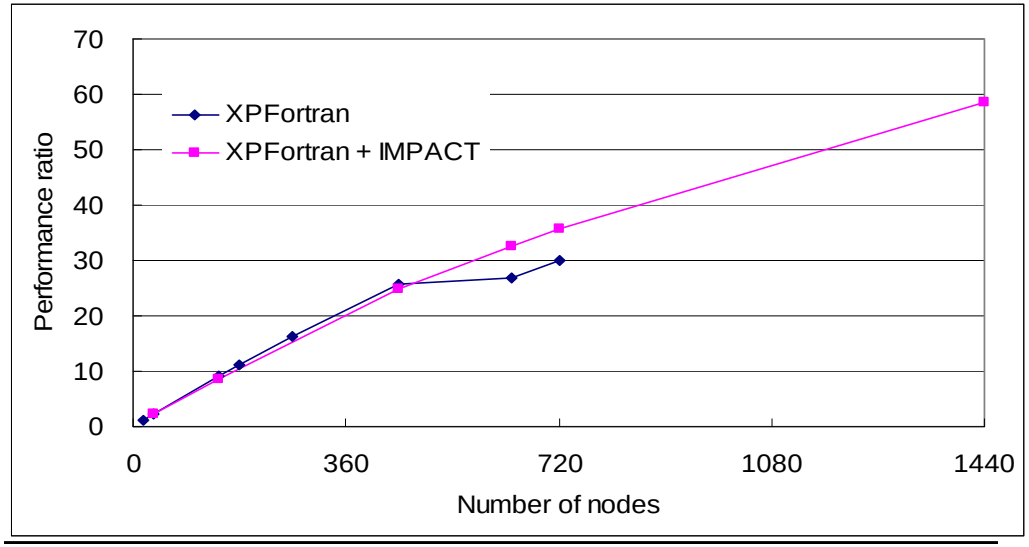
- データ同化
- PIV処理高速化
- システム開発，DB

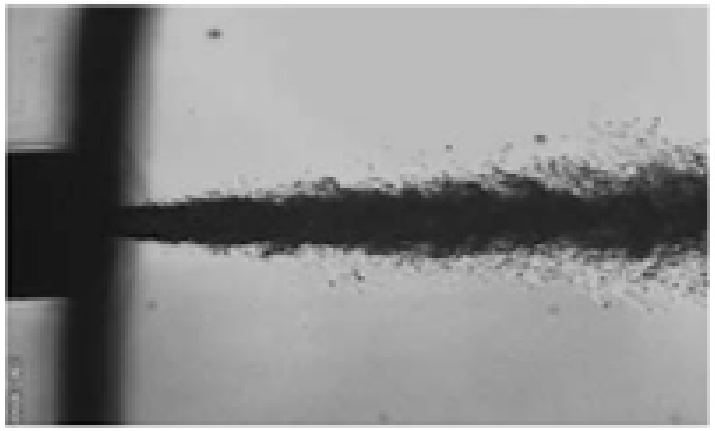
事例4： 発見的大規模解析

JAXA Supercomputer System

- 液体燃料噴流の微粒化過程の解明（LSC1）
  - 並列規模：1,440プロセス×4スレッド=5,760コア
  - 計算規模：（←計算時間ネック）
    - 格子点数：58億
  - 計算時間：410時間
  - 出力ファイル：153TB（25時間）
  - 実効効率：約4%程度

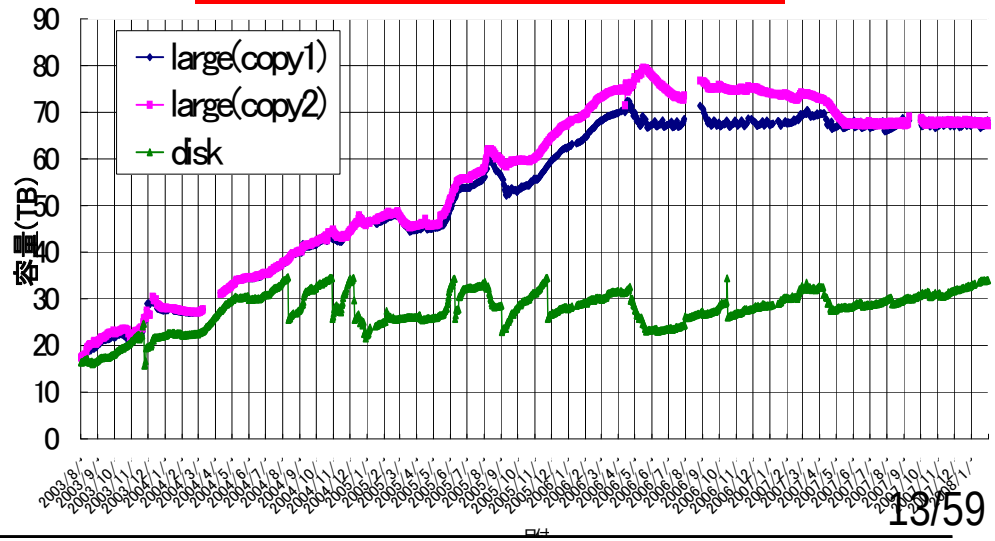




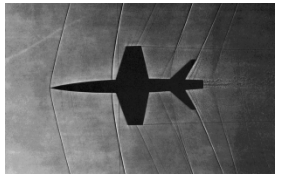
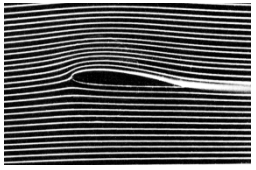


具体的に，どういう変化があったか

- 応用の幅が拡大
  - 要求の多様化，Capacityへの対応の必要性
  - F90，C/C++，構造体　→　コードの複雑化
  - PC, PCクラスタとの環境の連続性　→　VからSへ，機能性への要求
    性能以外への投資
- 工学系，プロダクションランの増加…多くのユーザとの信頼関係↑
  - 空力評価・設計探査（パラスタ），最適化　→　ケース数増大
  - 時間依存解析，アニメ作成　→　データ蓄積，セキュリティ
  - 業務コード，グループユーザ
- 制約・制限の増加
  - できれば良い
    → いついつまでにやれ
    費用対効果がどうの
    → オンディマンド



現状，何がネックか

- 格子生成
  - 境界層・衝撃波の存在，要求精度の高さ
  - 如何に実形状に短時間で対応するか　→　ワザが必要
- 並列化
  - マルチコア，マルチノード，プロセス並列，スレッド並列，・・・
    → 難しい印象，ユーザちゃんと理解していない，王道もなし
- データ処理
  - 単に可視化（表示，きれいな絵）すれば良いという時代は終わり
  - 多数データセット，時間依存データの処理
- 妥当性検証
  - 数学モデル，物理モデル，解法は完璧ではない
- 単体性能があまり上がらない
  - 数倍程度ではインパクトない（∵r^4）　→　パラダイムが描きにくい
  - 高くなる並列化の壁，使いにくいアーキテクチャ

今後の展望

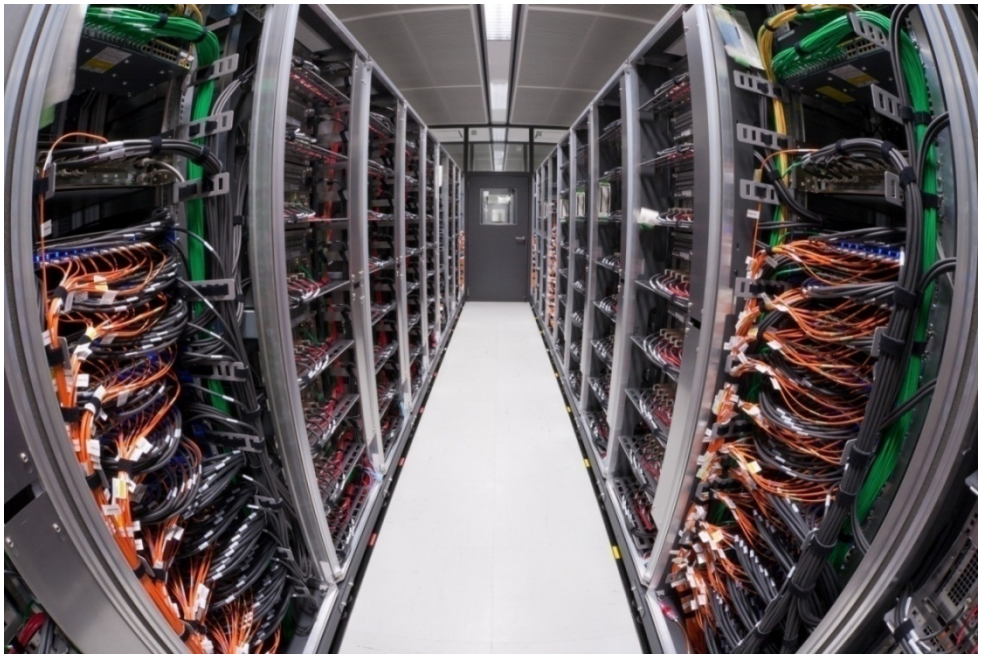
JAXA Supercomputer System

- この分野（航空宇宙，CFD）全体では，
  - 航空は成熟感，宇宙は意外に未整備
  - 分野融合，連成，最適化など応用の多様化進展
    - 「流体=ベクトル」という図式はもはや成立しない
    - メモリ性能の他の因子も考慮する必要あり
  - 市販コード，OSSの台頭
    - OpenFOAM（C++のCFDコード） → 市販コードなくなるかも
  - 時間依存解析，パラスタ解析の増加
    - データのさらなる増大，データマイニング技術など必要
- JAXAでは，
  - 実開発関連のジョブ（プロダクションラン）が相当数流れる
  - 単に大規模ではない ⇒ 業務で使う
    ⇒ 計算機規模は計算規模ではなくスループットで決まる
  - PCクラスタではできないことが明確化し，スパコンへ回帰
    - メモリ，ストレージ，運用管理

2. JAXAシステム(JSS)の概要

- 導入経緯，設計思想など
- システム概要・特徴

JAXA Supercomputer System: JSS



導入経緯

JAXA Supercomputer System

17年	平成18年度				平成19年度				平成20年度				平成21年度				平成22年度			
2006					2007				2008				2009				2010			
1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4

調達手続き

JAXA統合スパコン棟整備

（既存建屋活用分）

（新スパコン棟活用分）

ストレージ,A

P, V, L

M

JAXA統合スパコン
JSS1

2008年4月
導入開始

2009年4月
全体稼動

調布システム

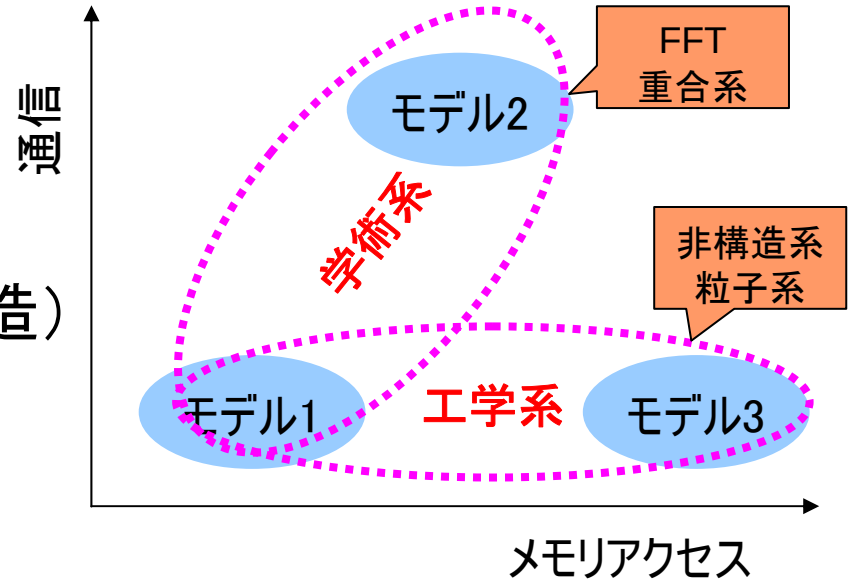
角田システム

相模原システム

JSS設計： アプリからの視点

- アプリモデル(1)
  - 工学系 ⇒ スループット重視（Capacity計算指向）
  - 学術系 ⇒ 性能，規模重視（Capability計算指向）

- アプリモデル(2)
  - モデル1：演算多（eg. ﾊﾟﾗｽﾀ）
  - モデル2：通信多（eg. FFT）
  - モデル3：メモリアクセス大（eg. 非構造）



どこを中心に設計するか？

JSS設計： 運用サイドからの視点

JAXA Supercomputer System

- 基本要件
  - 安定稼動（トラブルが少ない，設定が楽）
  - 運用管理が楽，運用コスト少
  - 設置性（スペース，電力，冷却）
  - ユーザに対して同一サービスを提供（公平性）
    - 遠隔からの利用に対しても
  - ソフトウェアの移植性，汎用性
  - システムの拡張性
  - 性能情報が取得可能なこと

システム導入における基本スタンス

JAXA Supercomputer System

- アプリありき，ユーザありき（凡庸でも）
  - 使ってなんぼの世界，業務の継続性は重要
  - アプリ，利用に対する明確なイメージ
- きちんと運用できないものは×
  - 可用性，信頼性
  - プロダクションラン増加への対応
  - F90，PC UNIX，市販アプリ等への対応
  - 利用における敷居を低く…移植性，利用環境の共通性
- （それなりに）業界を牽引
  - 三好さんの三位一体説（役所，ベンダー，ユーザ）踏襲
  - 周りの目，意地

JSSシステム構成

JAXA Supercomputer System

計算エンジン部

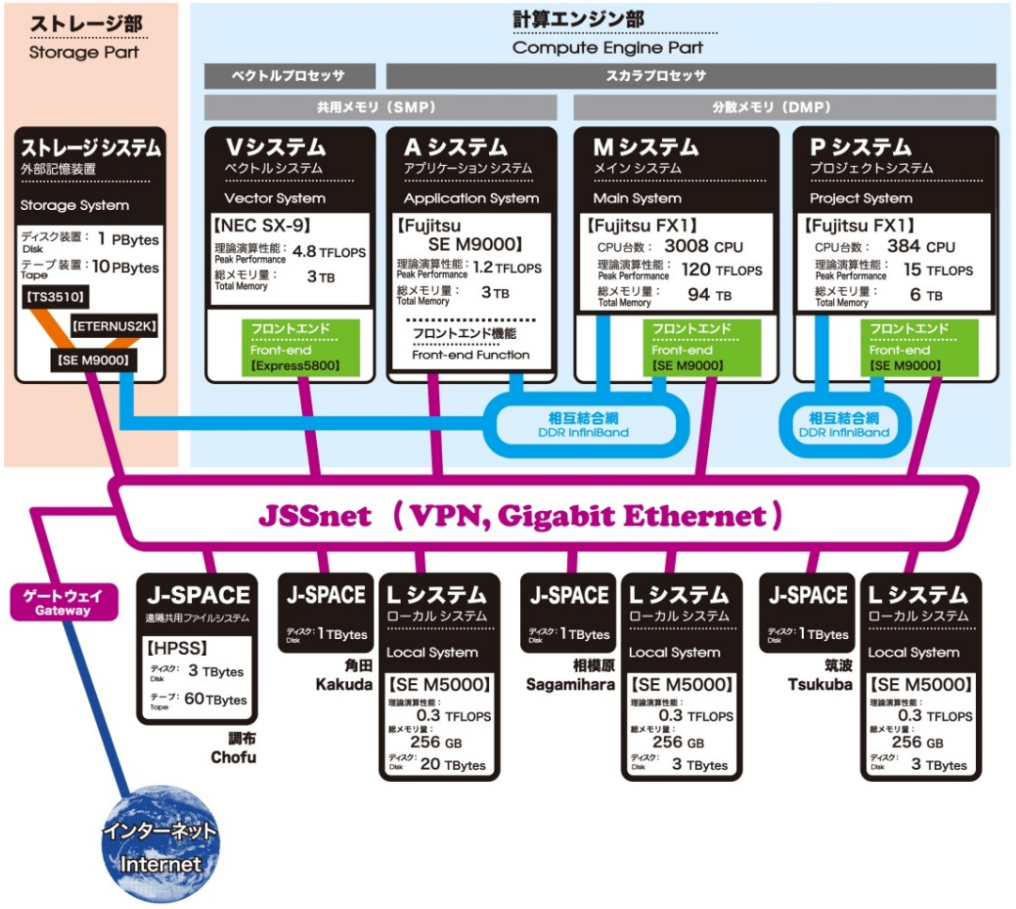
大規模並列計算機システム（M,P）
共有メモリ計算機システム（A,V）

ストレージ部

ストレージシステム

分散環境統合部

遠隔利用システム
分散データ共有システム
高速ネットワーク



計算エンジン部構成

システム名称	Mシステム	Pシステム	Aシステム	Vシステム
CPUタイプ	スカラー			ベクトル
システムタイプ	MPP		SMP	
ノード数	3,008	384	1	3
CPU数/ノード	1	1	32	16
コア数/CPU （全コア数）	4 （12,032）	4 （1,536）	4 （128）	1 （48）
ピーク性能[TFLOPS] （ノードあたり[GFLOPS]）	120 （40）	15 （40）	1.2 （1,200）	4.8 （1,600）
メモリ容量[TB] （ノードあたり [GB]）	94 （32）	6 （16）	1 （1,000）	3 （1,000）
製品名	富士通 FX1		富士通 SEM9000	NEC SX-9

JSSの特長

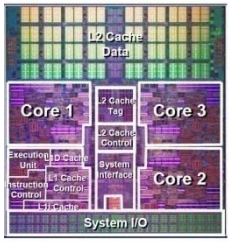
JAXA Supercomputer System

- 国内最高クラスの理論性能
  - スカラー：135TFLOPS，ベクトル：4.8TFLOPS
- 高い実行性能
  - 世界最高クラスのLINPACK性能：91.19%
- 実用計算志向，使い勝手・円滑な移行に配慮
  - 複数のアーキテクチャが混在＝選択の自由
  - 大規模メモリ：100TB以上
  - 大規模ストレージ：ディスク 1PB，テープ10PB
  - 共有メモリシステム：1TB共有メモリ
- 遠隔地からの利用環境
  - JSSネット：SINET3，VPNによる高速接続
  - ローカルシステム

計算エンジン部…JSS-M/P

JAXA Supercomputer System

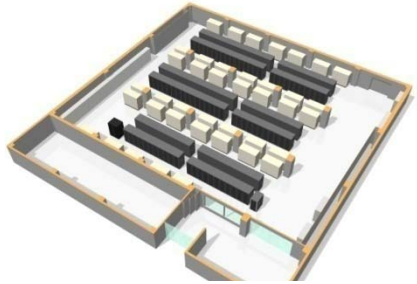
- 富士通製FX1クラスタ
- M: 3,008ノード（12,032コア）
  - QC SPARC64™ VII 2.5GHz
  - 40GFLOPS，32GB@ノード
- 94ラック
  - 32ノード，12KW@ラック
- FBBファットツリー・インターコネクト
  - DDR Infiniband
- M: 120TFLOPS，94TB
  - Linpack：110.6TFLOPS，91.19%
- JSS-Pはサブセット（15TF,384N）



1ノード@ボード

4ノード@シャーシ

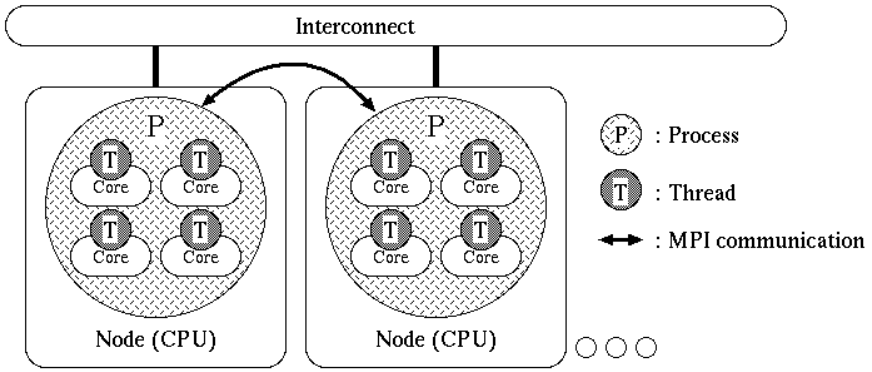
32ノード@ラック



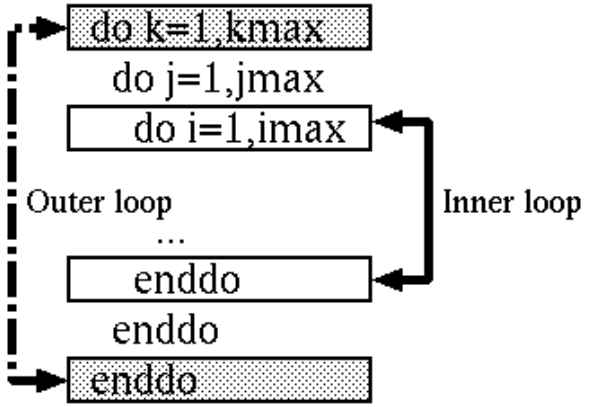
計算エンジン部…JSS-M/P（2）

JAXA Supercomputer System

- Integrated Multicore Parallel ArChiTecture：IMPACT
  - 6MB 共有L2キャッシュ
  - コア間ハードウェアバリア
  - 自動スレッド並列コンパイラ
    - 細粒度並列でも性能が出る



- 高メモリ性能
  - 高メモリバンド幅（40GB/s，B/F=1）
  - 低レイテンシ
  - 高信頼性（チップキル ECC）



- ノード間高速バリアネットワーク
  - データ転送とは別のNW
  - ノード間ハードバリア
  - 集合通信のHWサポート
  - OS割り込みによる遅延低減

計算エンジン部…JSS-A/V

- 1TBの大規模共有メモリマシン
- 前後処理，非並列ジョブ，特殊ジョブ（ベクトルジョブ，市販アプリ）
- JSS-A：富士通製SEM9000，1ノード
  - SPARC64TM VII 2.5GHz，4コア，40GF@ﾁｯﾌﾟ
  - 32CPU(128コア），1.2TFLOPS
  - Fluent, NASTRAN, FIELDVIEW

- JSS-V：NEC製SX-9，3ノード
  - 128GFLOPS@CPU
  - 16CPU，1.6TFLOPS ⇒ 4.8TFLOPS，3TB
  - ノード間はIXSで接続
  - ベクトルジョブ用

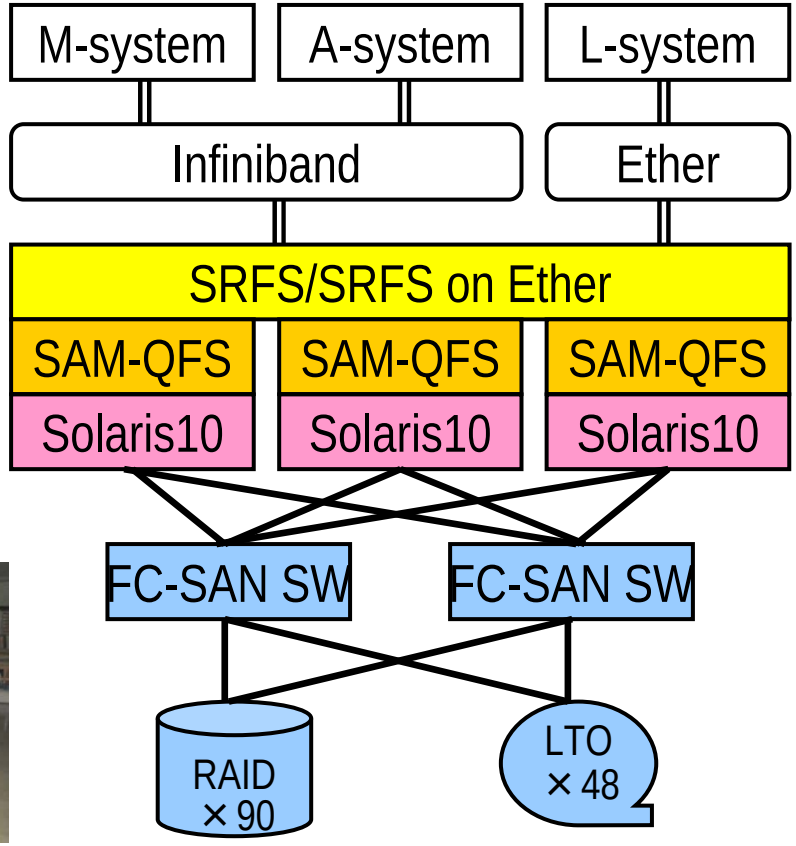
ストレージ部

- ディスク：1PB，F:ETERNUS2000
  - RAID5
  - 4Gbps FC：180本
  - キャッシュ：360GB
  - SATA：7200rpm，750GB

- テープ：10PB，IBM TS3500
  - 40×LTO4ドライブ，
    8×LTO3ドライブ
  - 4×TS3500ライブラリ，
    13,332カートリッジ

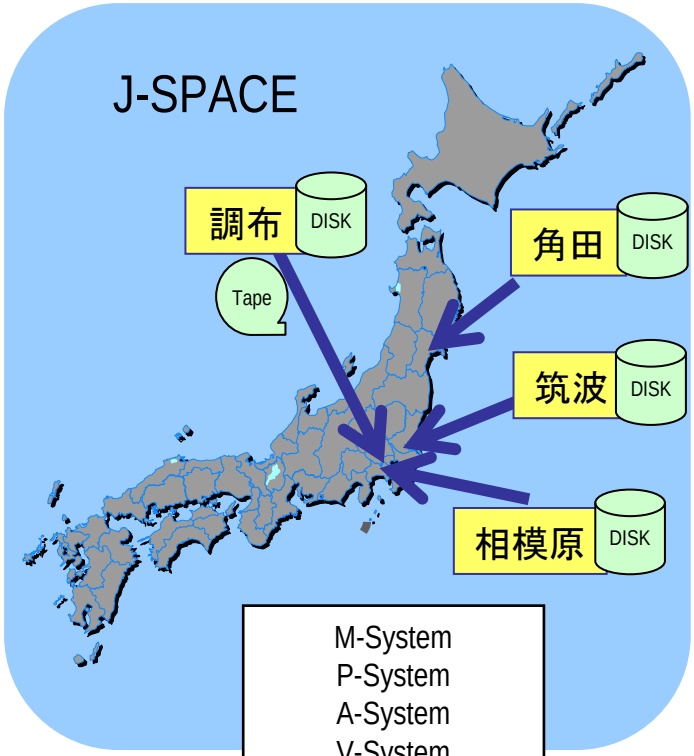
- I/Oサーバ：
  - F: SEM9000×3
- HSM：
  - S: SAM-QFS

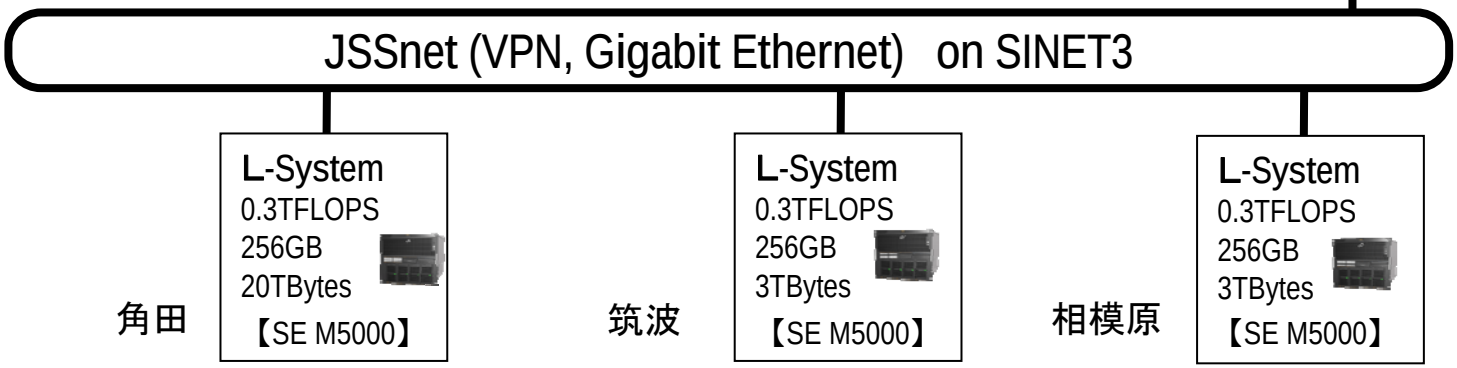
28GB/sのI/O性能
(ioperf)



分散環境統合部

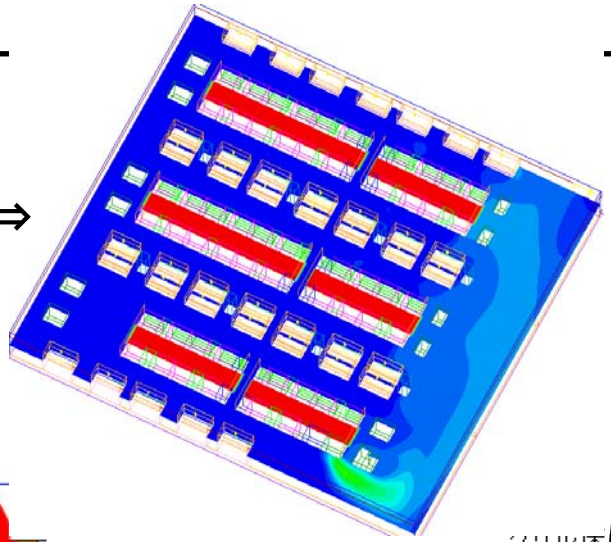
- 遠隔利用システム(Lシステム):主要事業所へのフロントエンド機能の提供
  - 角田，筑波，相模原
- インターネット(SINET3)越しの高速なファイル共有
  - SRFS on Ether
- 各拠点間でのデータ共有が可能な分散データ共有システム
  - J-SPACE(HPSS)

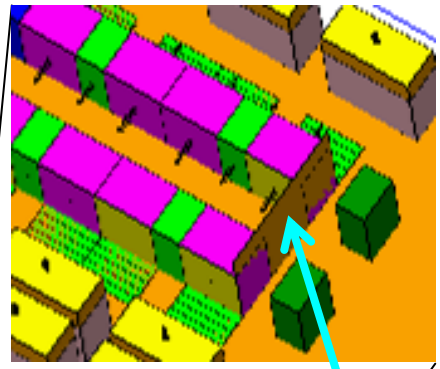




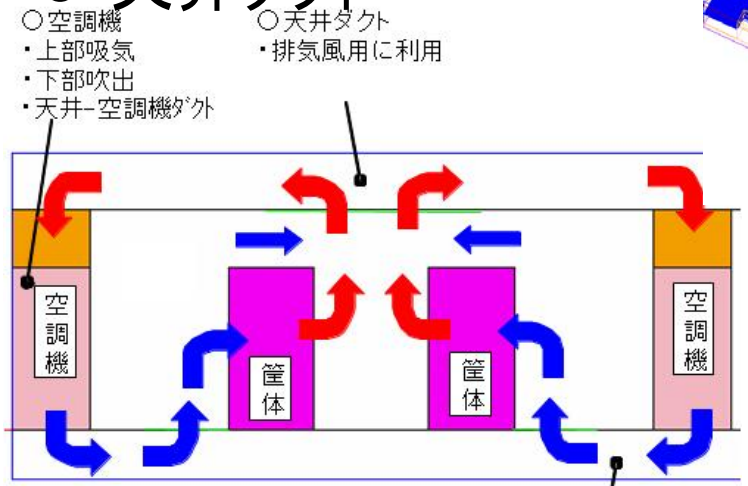
冷却対策

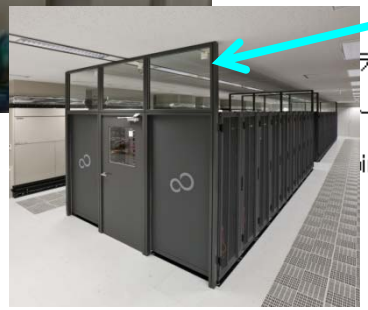
- 排気拡散防止板
  - CFDにより解析・予測 ⇒





- 天井ダクト





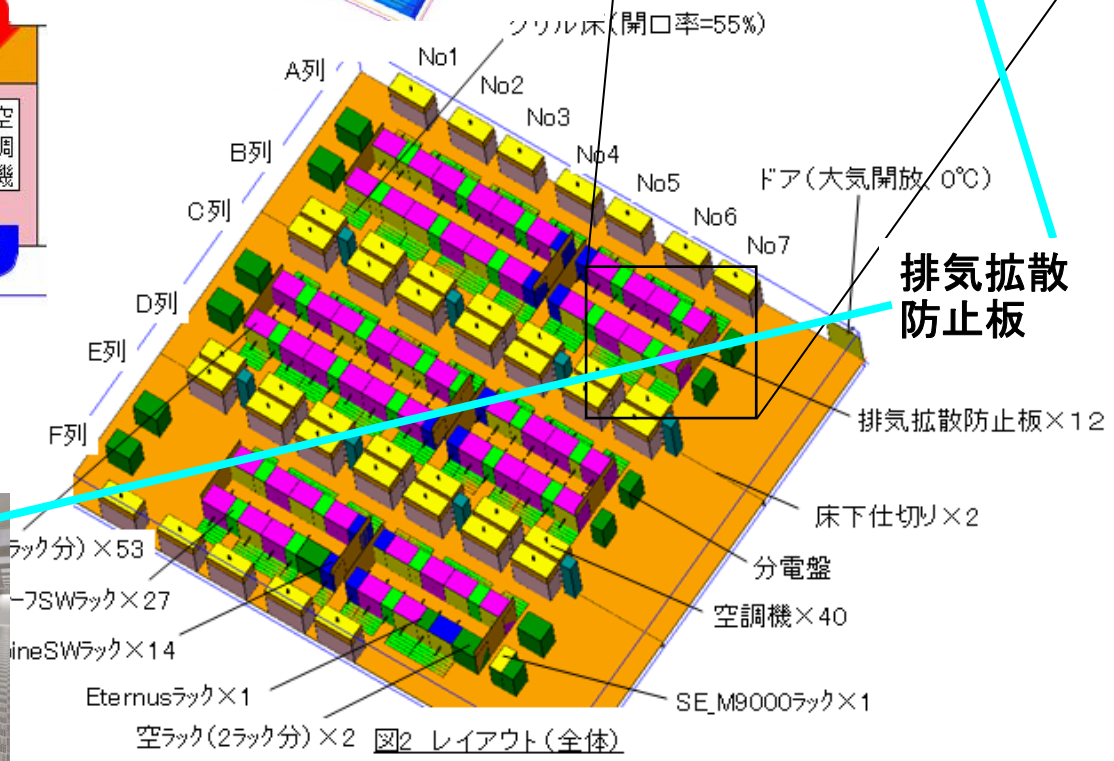


図2 レイアウト（全体）

Linpack HPL

JAXA Supercomputer System

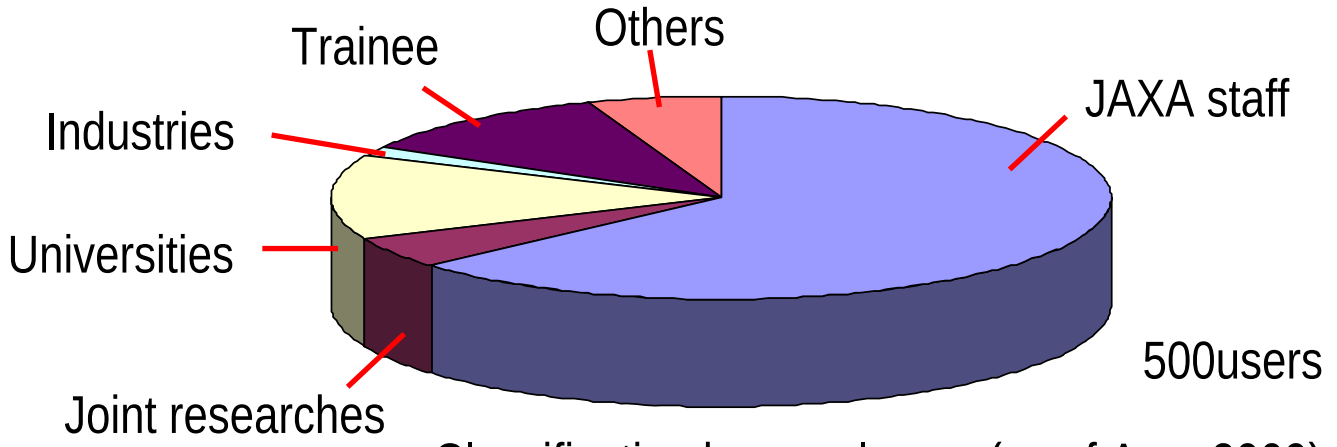
- 我々の考えるLinpackベンチの意味
  - 全システム稼動確認，データ取得　→　動作状態，電力，空調
  - 実行効率の目安　　91.19%
  - 耐久試験　　60時間40分

Top500ランキング（国内分）

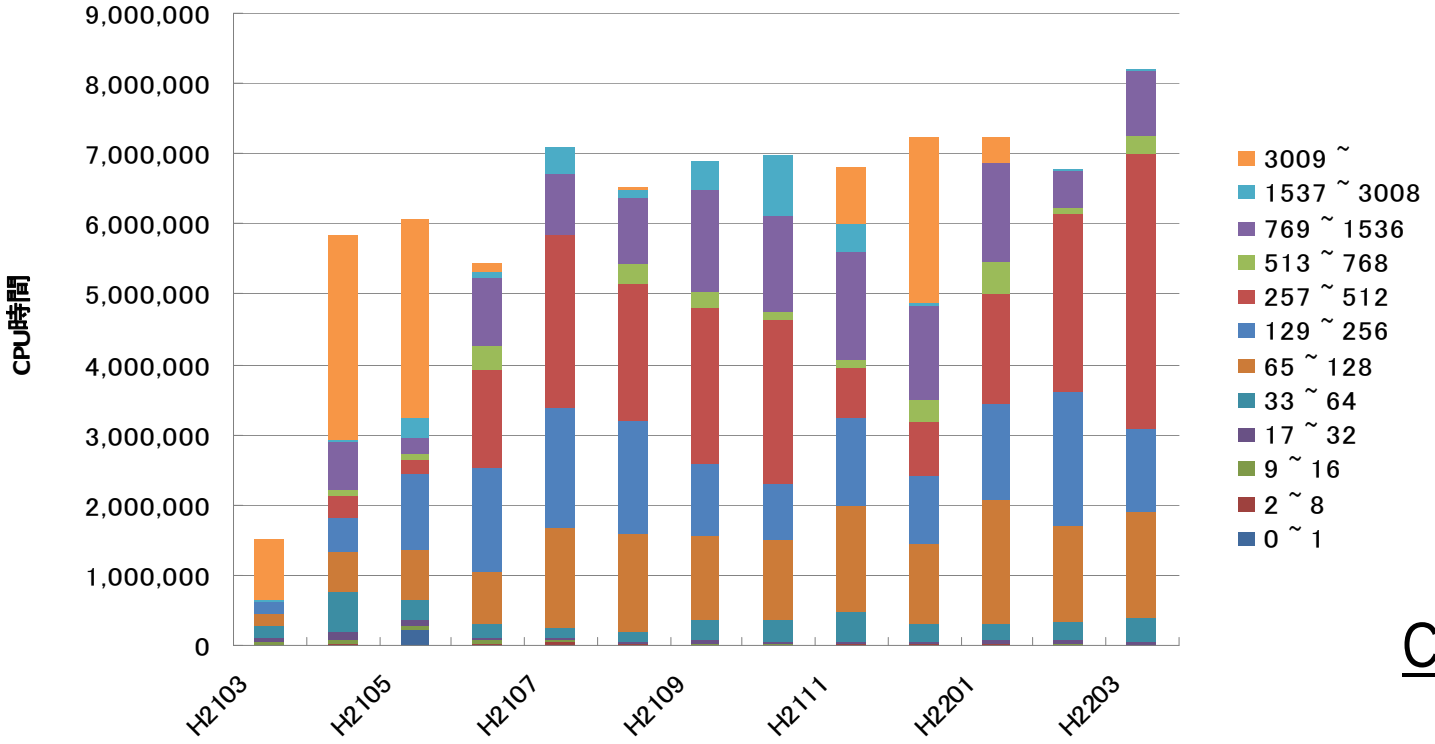
順位	サイト	マシン	コア数	Rpeak [TFlops]	Rmax [TFlops]	効率 [%]
22	地球シミュレータ	SX-9/E	1,280	131,072	122,400	93.38
28	JAXA	FX1	12,032	121.282	110.600	91.19
40	理研	RX200S5	8,256	96,760	87,890	90.83
41	東工大	Sun Fire	31,024	163,188	87,010	53.32
42	東大	HA8000	12,288	113,050	82,984	73.40
47	筑波大	Xtreme-X3	10,368	95,385	77,280	81.02

［2009.6現在］

システム利用実績



Classification by user layers (as of Aug, 2009)



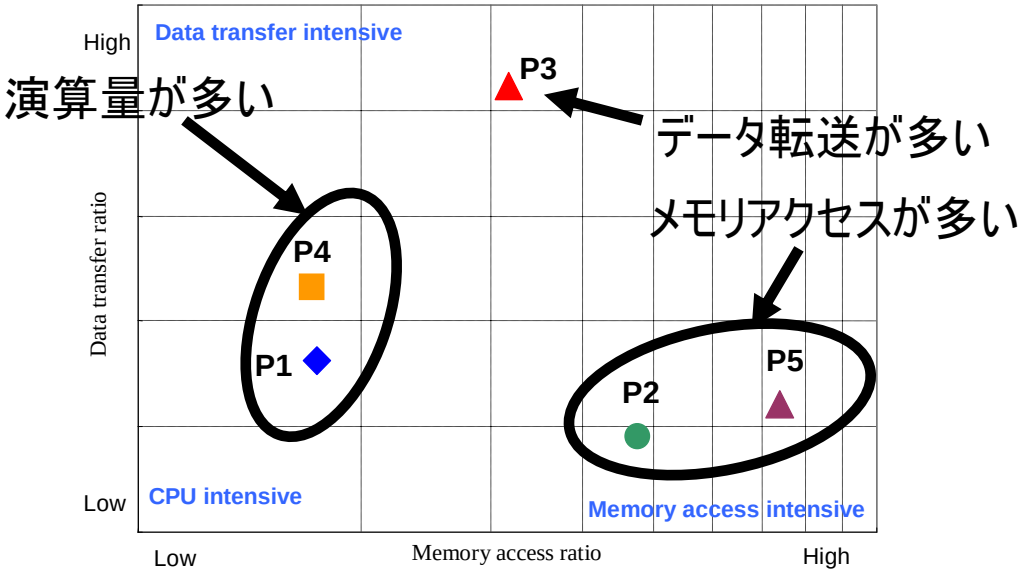
CPU usage

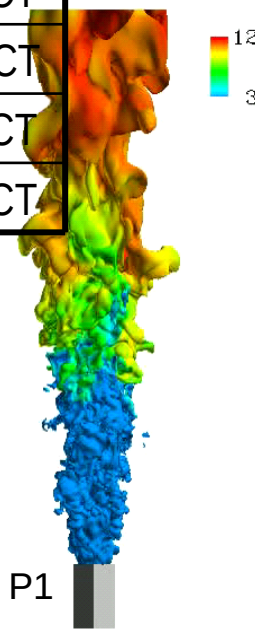
3. JAXAアプリとその性能チューニング

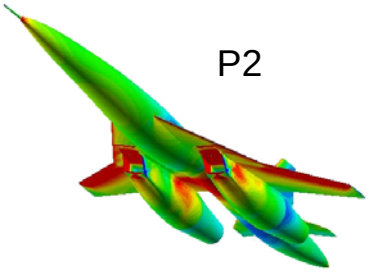
JAXA Supercomputer System

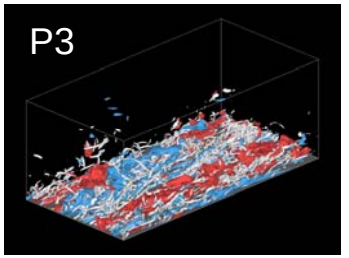
- JAXAの代表的なアプリケーション(コード)

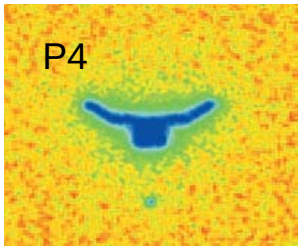
ｺｰﾄﾞ名称	対象	手法	特徴	並列化
P1	燃焼	FDM+化学反応	DIV,SQRTループ	MPI+IMPACT
P2	汎用	FVM(構造)	ポインタ配列	MPI+IMPACT
P3	乱流	FDM+FFT	コピールーチン	XPF+IMPACT
P4	プラズマ	PIC	粒子系	MPI+IMPACT
P5	汎用	FVM(非構造)	リストアクセス	MPI+IMPACT

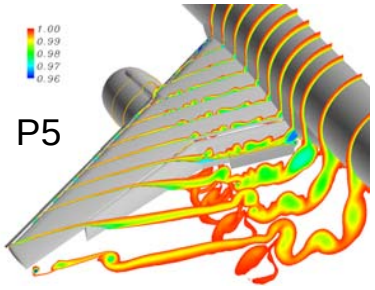










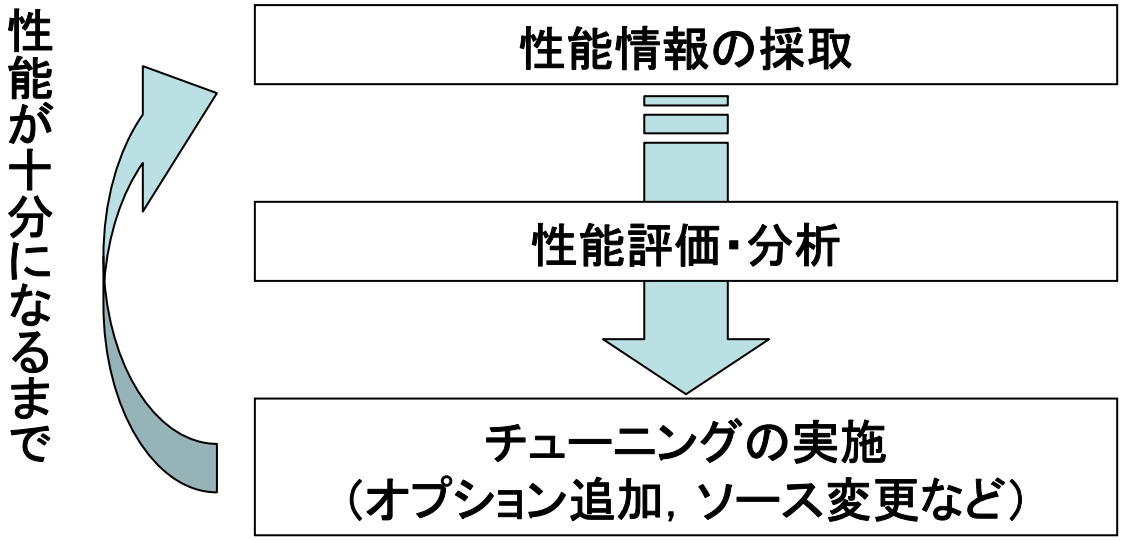


性能チューニングとは

JAXA Supercomputer System

- そのマシンの性能を（最大限に）引き出す
  - アプリケーションによりプログラムの性質が異なる（特性，書き方）
  - 理論性能 vs 実行性能
- チューニングが必要な場合
  - 計算時間を短くしたい
  - プラットフォームが変わった
  - コードを作り直した
  - 問題規模を変えたい
- ぎりぎりのチューニング
  - とにかく性能を最大に，プログラムが見難くなっても良い
- 一般のチューニング
  - ユーザが納得・了解の範囲
  - 移植性が重要
  - 継続的にプログラムの発展・拡張が可能

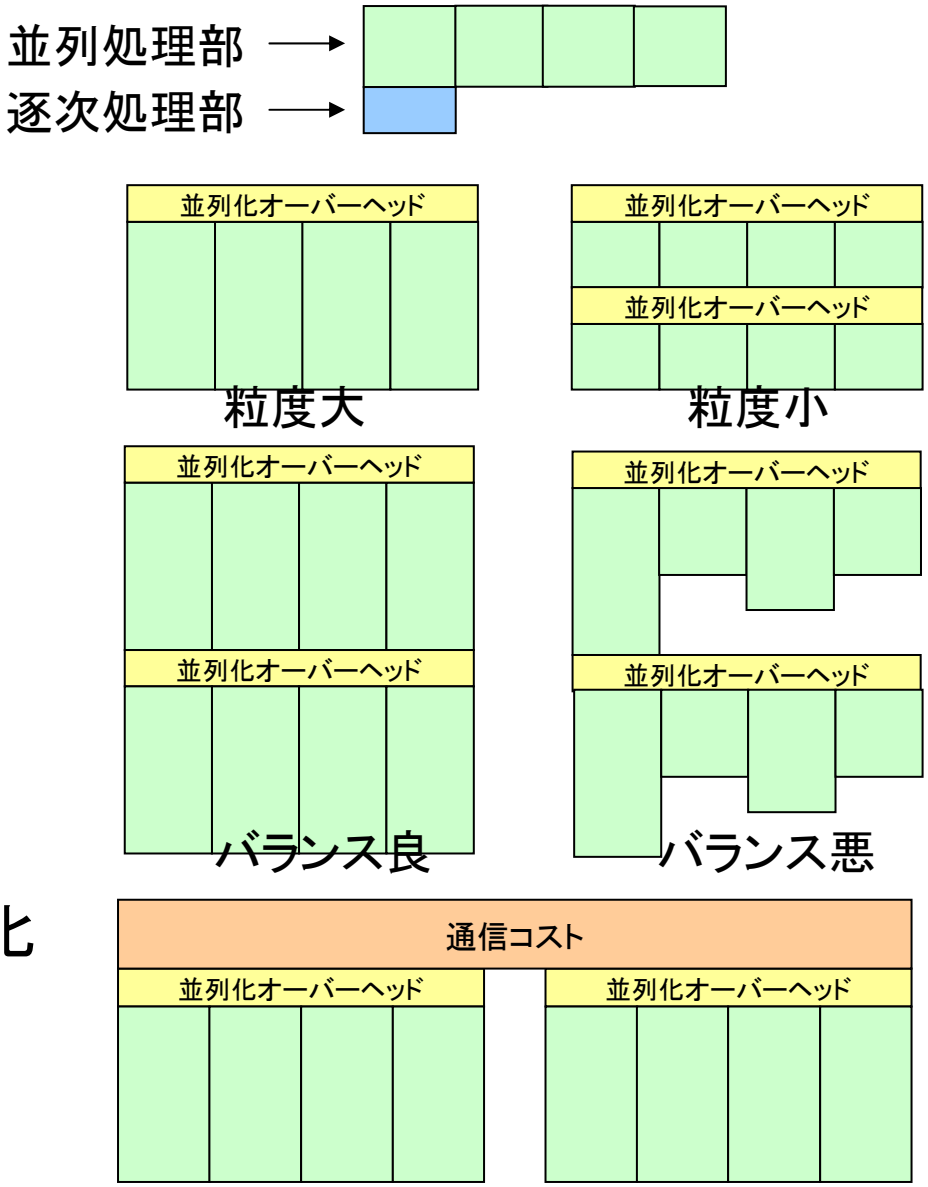
性能チューニングの流れ



1. 性能情報の取得
   - 性能分析ツール（プロファイラ）を使用し，性能情報を採取
2. 性能の評価・分析
   - 性能の目安値と性能情報を比較し，不十分ならばソースやプロファイラ情報から原因を分析
3. チューニングの実施
   - 翻訳/実行オプションを追加，ソースコードの変更

性能を阻害する要因

- 並列化率
  - アムダールの法則
- 並列化粒度
- ロードバランス
- 通信コスト
- キャッシュ競合
  - 最低8個データが離れていないと同じデータを参照
    ⇒　外側ループで並列化



知っておくべきこと

JAXA Supercomputer System

- Strong Scaling，スピードアップ並列性能
  - 問題規模一定，処理のオーバーヘッド，通信レイテンシ
  - アムダールの法則

$$S(n) = \frac{1}{(1-a) + a/n}$$

    - 逐次処理部分が10%あれば，S(∞) = 10: 性能向上率は高々10倍

- Weak Scaling，スケールアップ並列性能
  - CPU負荷一定，メモリ性能，通信スループット
  - グスタフソンの法則

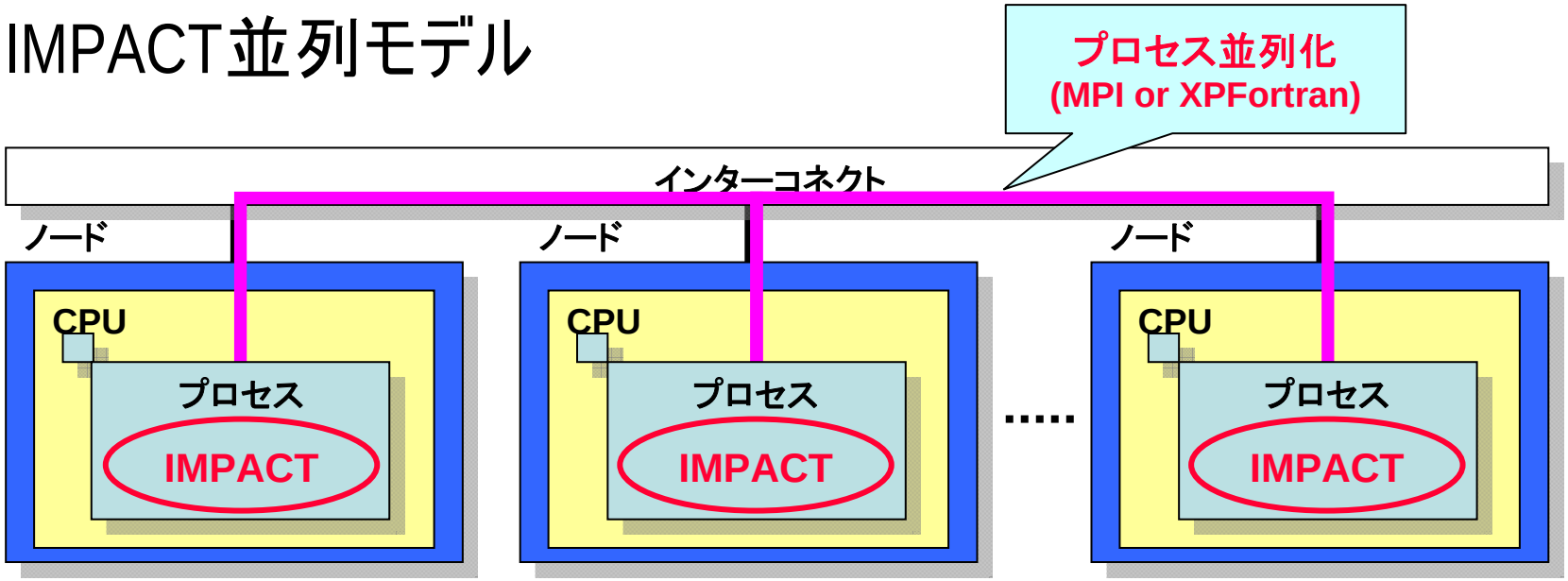
$$S(n) = 1 + a(n-1)$$

    - 90%の並列化率があれば，S(n) = 0.9n : 性能向上率はほぼ比例

JSSの並列プログラミングモデル

JAXA Supercomputer System

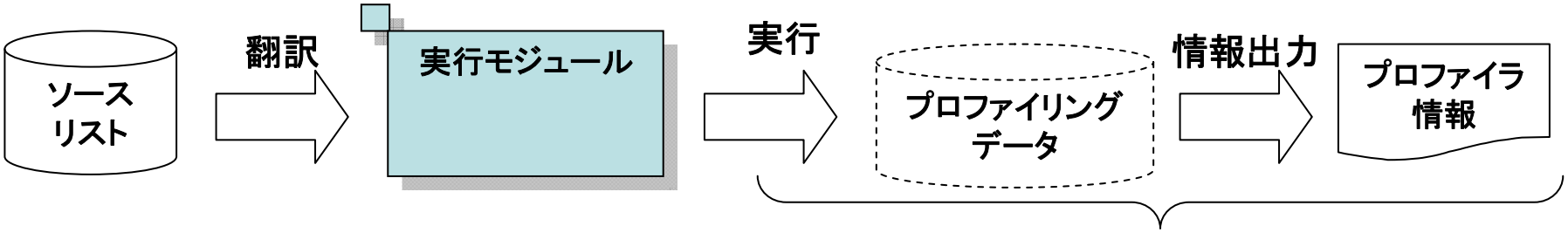
- IMPACT並列モデル



- プログラムはプロセスを単位として動作.
- プロセス内部はIMPACTにより処理を自動並列化して実行.
- また，MPIやXPFortranを使用すると，複数のプロセスに分割して実行することができる（これをプロセス並列化という）.
- IMPACTとプロセス並列化を上図のように組合わせることで，複数ノードを使用する大規模なプログラムを実行することが可能

プロファイラの概要

- プロファイラとは
  - プログラムの性能分析を定量的に行なうために必要な情報を収集するツール
- 機能概要
  - 経過時間，CPU時間情報
  - プロセス間通信の時間情報
  - サンプリングによるコスト分布情報
  - 実行時のハードウェア操作状況
- 動作概要



プロファイラによる情報採取手順

① **翻訳**…情報収集対象プログラムをコンパイラで翻訳

```
f90jx    sample.f90  -o  sample.exe
```

② **実行とプロファイラ情報の出力**…出力ファイル名を実行モジュール名を指定してプロファイラコマンド(*fpcoll*)を実行するとプログラム実行と同時に情報収集を行い結果をファイルに出力

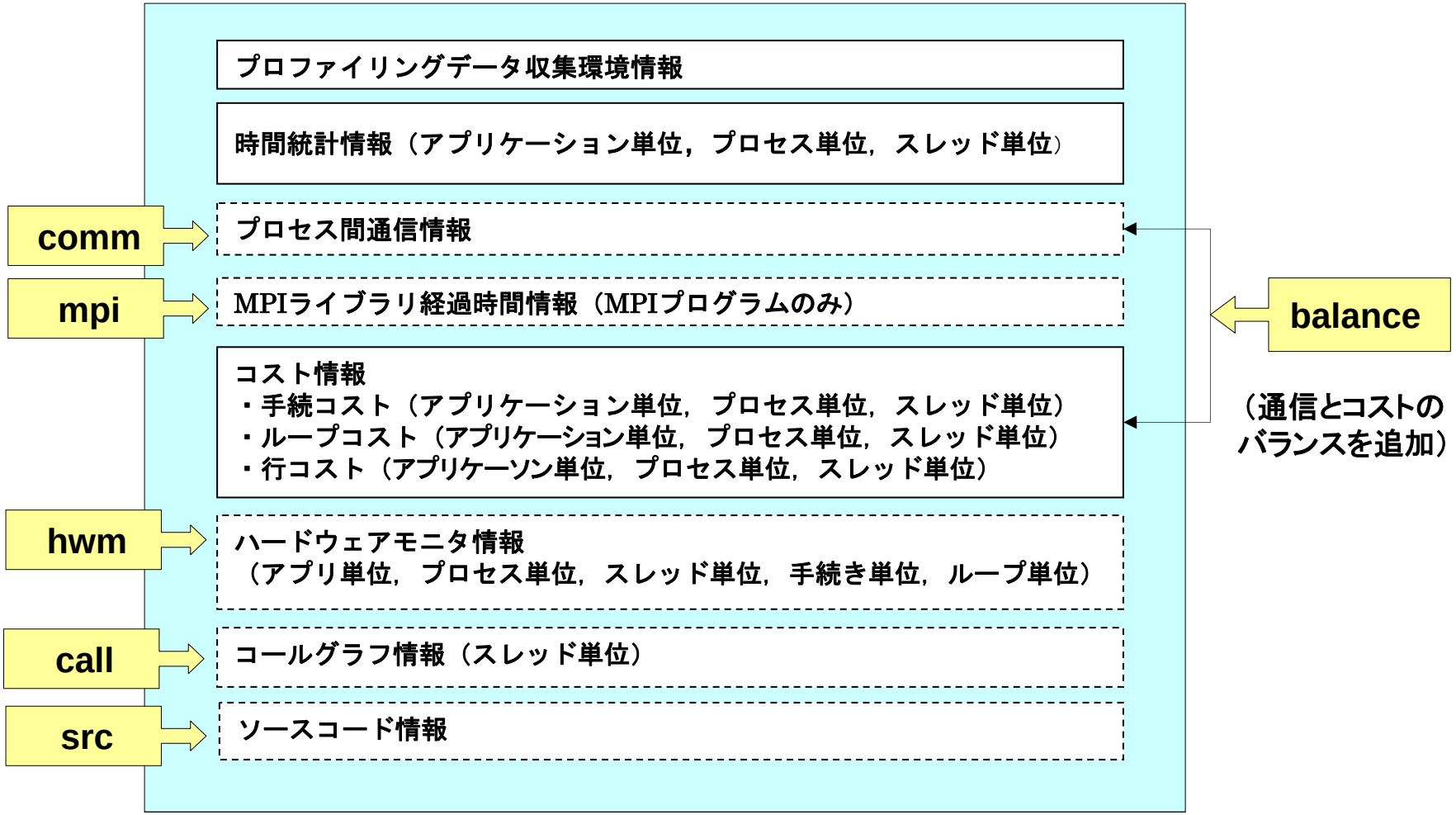
```
fpcoll -Icall,balance,hwm -I30 -o result.txt ./sample.exe
```

- 収集および出力項目
- 手続情報の出力件数
- プロファイラ情報出力ファイル名
- ①で生成した実行モジュール

プロファイラ採取情報の見方(1)

JAXA Supercomputer System

- Fpcollコマンドで出力されるプロファイラ情報の構成



※ 黄色箱は，出力項目を指定するオプション．実線内は標準出力

プロファイラ採取情報の見方(2)

JAXA Supercomputer System

- 性能チューニングの指標となる代表的な性能情報

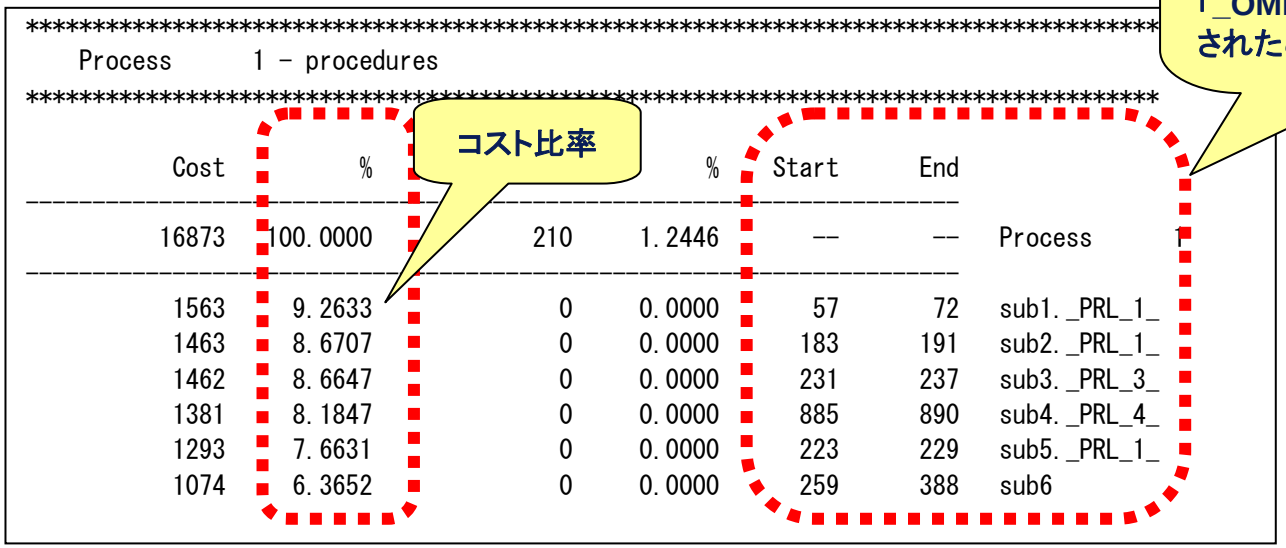
性能情報	内容	性能目安
MIPS値	命令の実行効率 (命令実行数÷CPU時間÷1e+6)	4,000以上 (1プロセスあたり)
MFLOPS値	浮動小数点命令実行速度 (浮動小数点演算命令数÷CPU時間÷1e+6)	2,000以上 (1プロセスあたり)
L2キャッシュミス率	2次キャッシュのミスヒット率 (2次キャッシュミス回数÷命令実行数)[%]	0.2%未満 (1プロセスあたり)

  - これらの情報はハードウェアモニタ情報から参照可能
  - 出力値が性能目安に満たない場合は要チューニング

スカラー性能の評価(1)

JAXA Supercomputer System

- コスト情報　⇒　ホットスポットの抽出




```
*******************************************************************************
    Process       1 - procedures
*******************************************************************************

          Cost       %          %  Start      End
-------------------------------------------------------------------------------
         16873  100.0000      210  1.2446     --       --    Process       1
-------------------------------------------------------------------------------
          1563    9.2633        0  0.0000     57       72    sub1._PRL_1_
          1463    8.6707        0  0.0000    183      191    sub2._PRL_1_
          1462    8.6647        0  0.0000    231      237    sub3._PRL_3_
          1381    8.1847        0  0.0000    885      890    sub4._PRL_4_
          1293    7.6631        0  0.0000    223      229    sub5._PRL_1_
          1074    6.3652        0  0.0000    259      388    sub6
```

- コスト比率が高いところは，全体時間に対して多くの実行時間を要する箇所(=ホットスポット)になっている.
- コスト情報の上位に現れる手続き名とその行番号に注目

スカラー性能の評価(2)

JAXA Supercomputer System

- ハードウェアモニタ情報 ⇒ 性能指標の判定

MIPS値

MFLOPS値

```
**********************************************************************************************
    Process      1 - performance monitors
**********************************************************************************************

                 Time(s)   Instructions          MIPS        MFLOPS     Cover(%)   Start      End
----------------------------------------------------------------------------------------------
   Elapsed       83.5100   292547367851     3503.1428      770.6741      98.4849      --       --   Process      1
----------------------------------------------------------------------------------------------
   Elapsed        4.3556    26445838686     6071.6699     1470.7395      98.2867      57       72   sub1._PRL_1_
   Elapsed        5.2387    23881767634     4558.6968      824.4698      98.3642     183      191   sub2._PRL_1_
   Elapsed        4.0266    20498623749     5090.8169     1259.5062      98.3919     885      890   sub3._PRL_4_
   Elapsed        3.6865    18813359810     5103.2896     1332.8138      98.7677     231      237   sub4._PRL_3_
   Elapsed        3.6244    18026236014     4973.6074     1264.7723      98.6891     223      229   sub5._PRL_1_
   CPU           11.2920    34802192952     3082.0183      580.0624      98.9115     259      388   sub6
```

L2キャッシュミス率

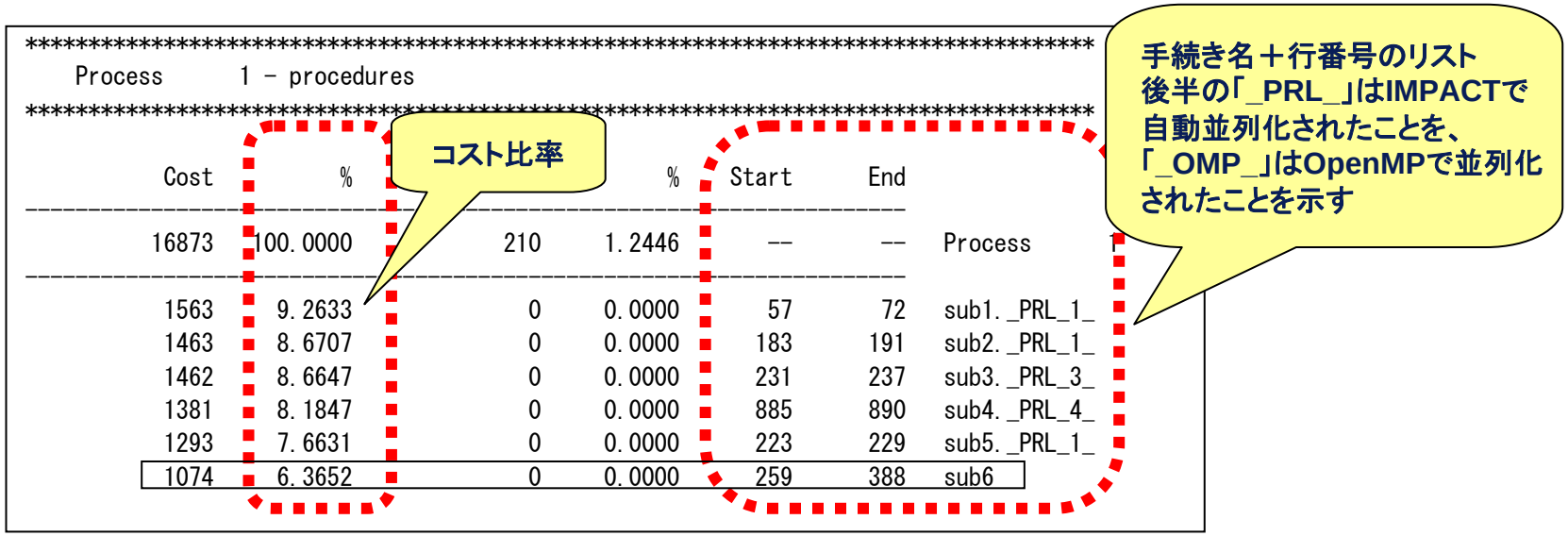
性能目安に満たないものがチューニング対象になる

```
                 Time(s)     L2 miss(%)     mTLB-is(%)     mTLB-op(%)     Cover(%)   Start      End
----------------------------------------------------------------------------------------------
   Elapsed       83.5100         0.3323         0.0000         0.0000      98.4849      --       --   Process      1
----------------------------------------------------------------------------------------------
   Elapsed        4.3556         0.3377         0.0000         0.0000      98.2867      57       72   sub1._PRL_1_
   Elapsed        5.2387         0.4233         0.0000         0.0000      98.3642     183      191   sub2._PRL_1_
   Elapsed        4.0266         0.2947         0.0000         0.0000      98.3919     885      890   sub3._PRL_4_
   Elapsed        3.6865         0.2825         0.0000         0.0000      98.7677     231      237   sub4._PRL_3_
   Elapsed        3.6244         0.2875         0.0000         0.0000      98.6891     223      229   sub5._PRL_1_
   CPU           11.2920         0.3243         0.0000         0.0000      98.9115     259      388   sub6
```

自動並列性能の評価(1)

JAXA Supercomputer System

- コスト情報　⇒　自動並列化されていない手続きの抽出

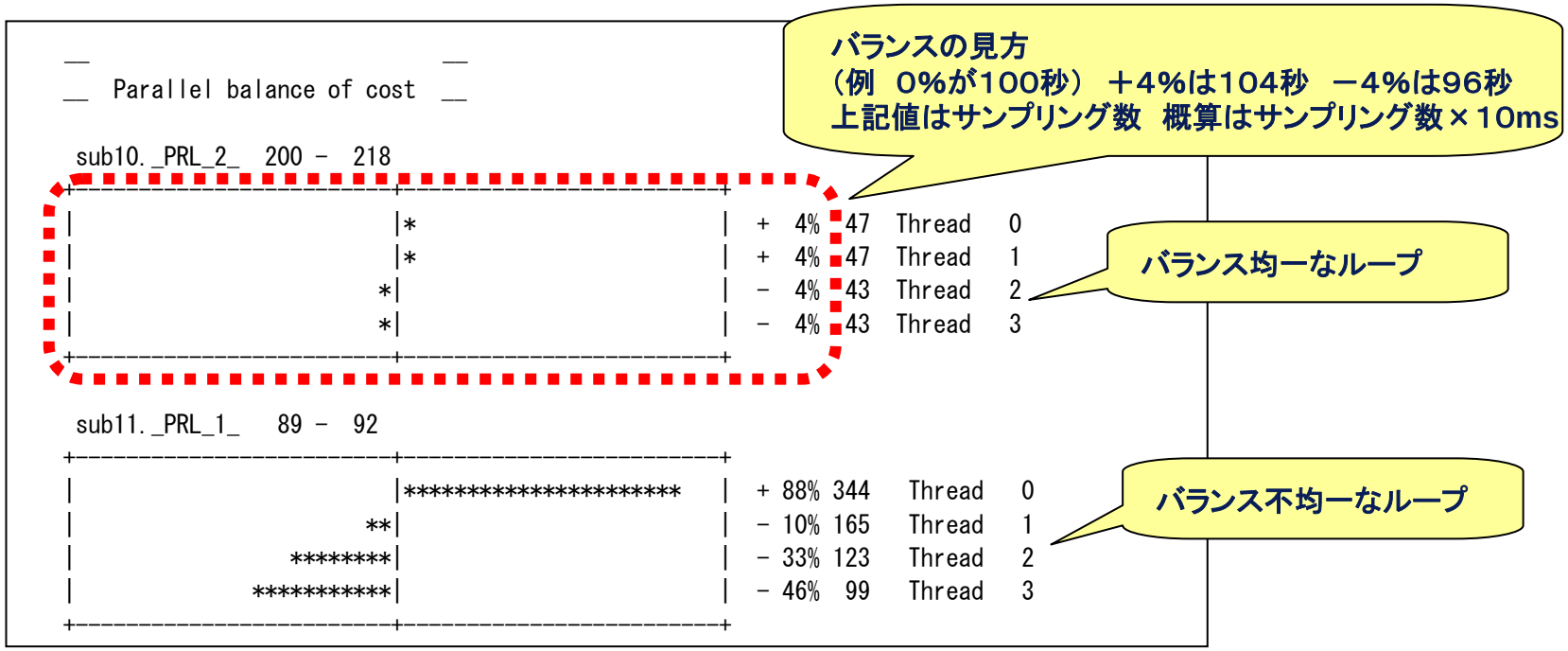


- 手続き名に「_PRL_」が付いていないものは自動並列化が阻害されている可能性がある．コスト情報の上位に現れる自動並列化されていない手続き名とその行番号に注目．

自動並列性能の評価(2)

JAXA Supercomputer System

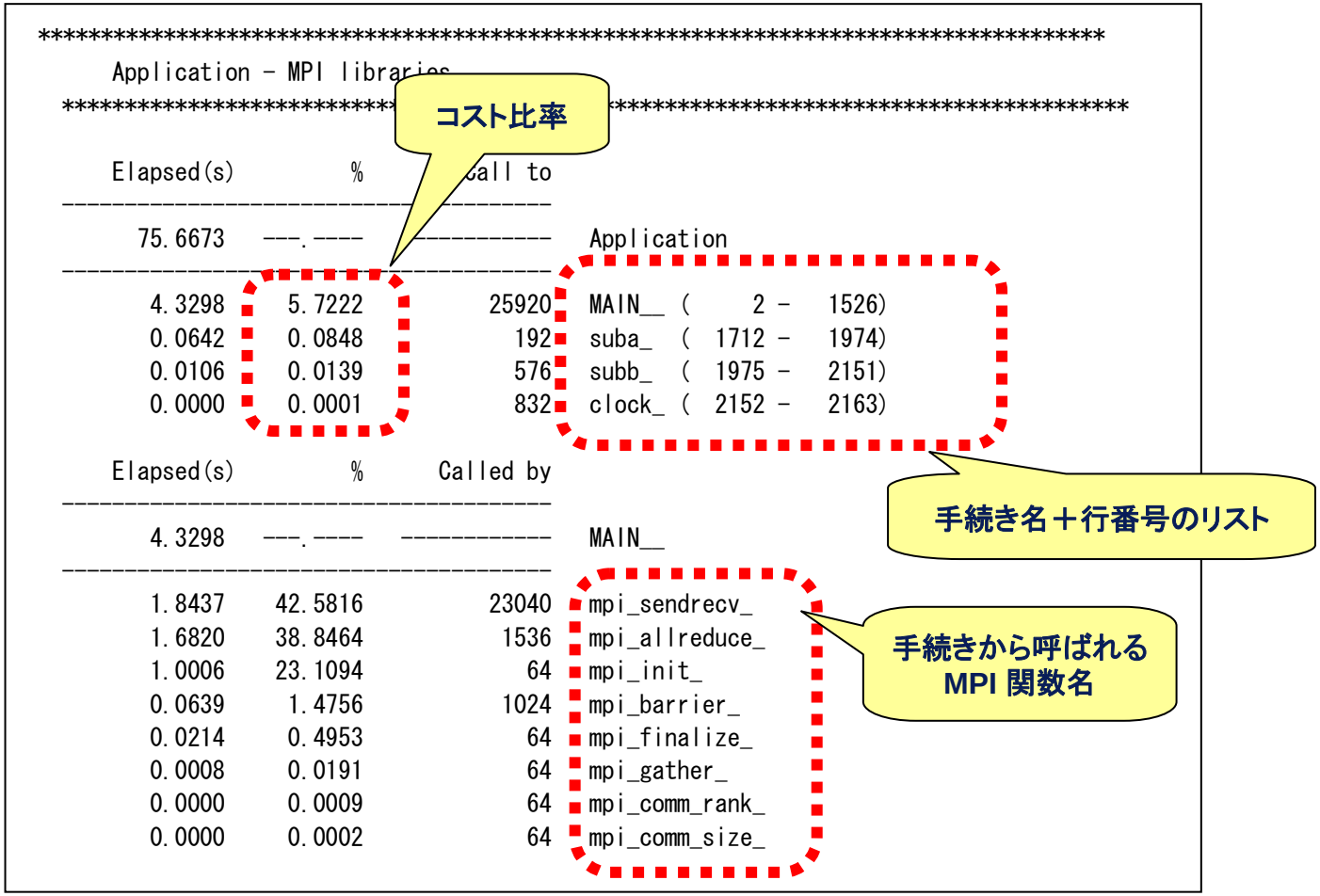
- バランス情報　⇒　ロードバランス不均一な手続きの抽出



- 自動並列化された手続きでも，ロードバランスが不均一なために十分な並列効果が出ない場合がある
- バランス情報にスレッド単位のコストが表示されるので，コスト情報の上位に現れるバランス不均一な手続きとその行番号に注目

MPI並列性能の評価

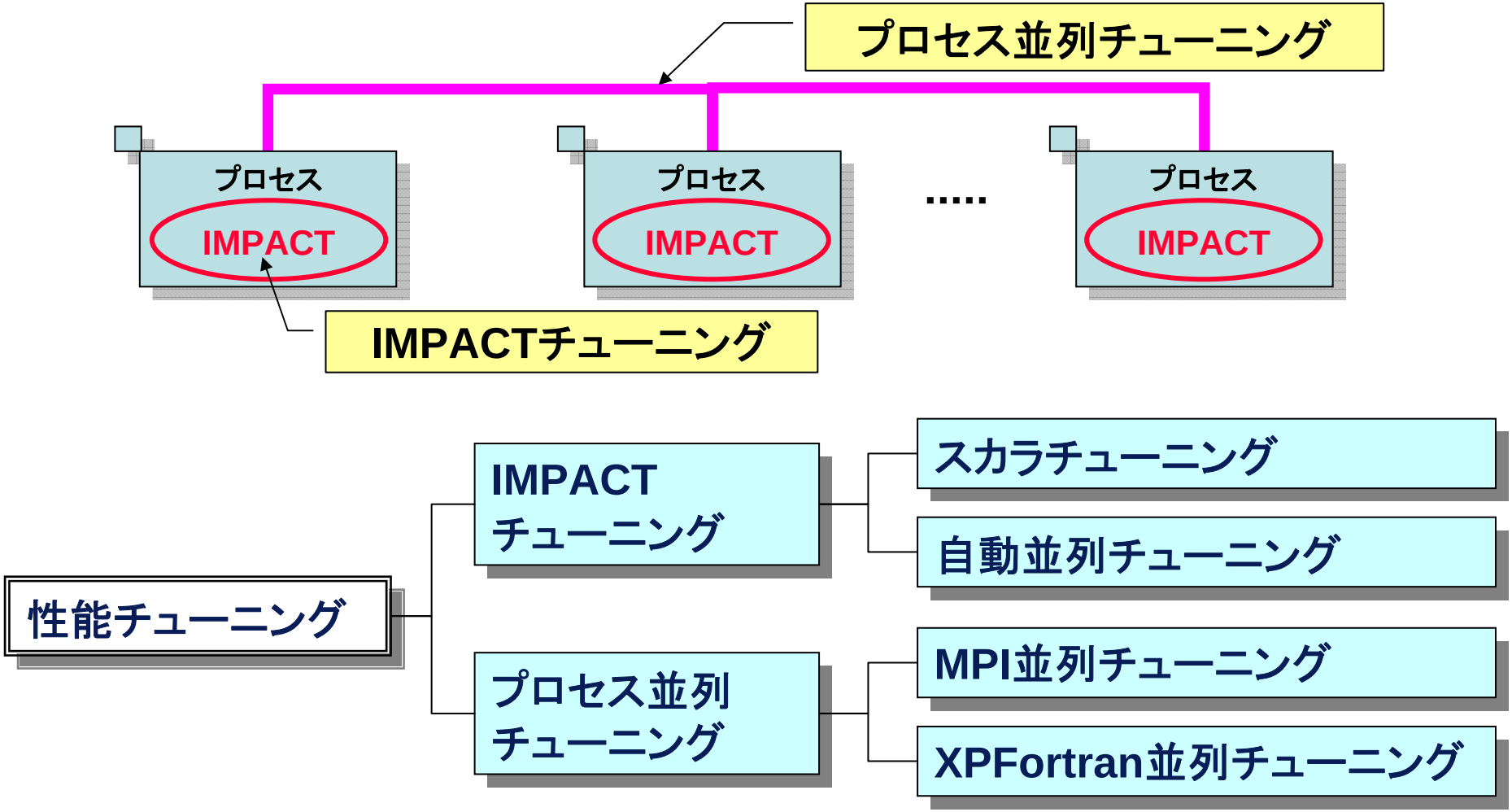
- MPIライブラリ経過時間情報　⇒　MPIライブラリ情報抽出



JSSにおける性能チューニングの概要

JAXA Supercomputer System

- プログラミングモデルのどの部分を対象とするかで分かれる



性能チューニングのポイント

JAXA Supercomputer System

- スカラーチューニング
  - データの局所性を高める
  - 演算器の実行効率を高める
- 自動並列チューニング
  - 並列化率を上げる
  - 並列化粒度を上げる
  - ロードバランスを均一化させる
- MPI並列チューニング
  - ロードバランスを均一化させる
  - プロセス間の通信コストを削減する
- XPFortran並列チューニング
  - 並列化率を上げる
  - 並列化粒度を上げる
  - ロードバランスを均一化させる
  - プロセス間の通信コストを削減する

スカラー…オプションチューニング

JAXA Supercomputer System

- 標準よりも強力な最適化を指示するコンパイラ・オプションを利用

チューニング内容	対象となる翻訳オプション	最適化の効果
配列のパディング	-Karraypad_const -Karraypad_expr	キャッシュ競合の削減
配列の次元入替え	-Karray_subscriptなど	キャッシュ利用効率の向上
配列の融合	-Karray_mergeなど	キャッシュ利用効率の向上
インダイレクトアクセスのプリフェッチ	-Kprefetch_indirect	メモリアクセスの高速化
IF構文に含まれるアクセスのプリフェッチ	-Kprefetch_conditional	メモリアクセスの高速化

  - 効果の有無はプログラム・データ特性によって変わるので，実際にオプションを付けて翻訳した実行モジュールで確認する必要がある

自動並列… コンパイルリストの利用

JAXA Supercomputer System

- 診断メッセージ ⇒ 翻訳時に状況をメッセージ出力

翻訳時オプション形式： -Kpmsg[= { 1 | 2 | 3 }]　　(Fortran/C共通)

オプション	機能
-Kpmsg=1	自動並列化した旨のメッセージのみ出力
-Kpmsg=2	-Kpmsg=1に加え、自動並列化できなかったことを示す簡略化メッセージを出力
-Kpmsg=3	-Kpmsg=1に加え、自動並列化できなかったことを示すメッセージを出力

- 翻訳例，診断メッセージ出力

```
$ f90jx –Kpmsg sample.f90
```

```
Diagnostic messages: program name(sample)

jwd5143i-i  “sample.f”, line 406: DOループの繰返し数が少ないため,
このDOループは並列化されません.

jwd5001i-i  "sample.f", line 695: このDOループは,並列化されました.
(名前:j)
```

自動並列… OCL行の挿入

JAXA Supercomputer System

- 自動並列化促進のためコンパイラに指示を与える

```
!OCL INDEPENDENT(SUB)
      DO I=1,N
         CALL SUB(A(I))
      ENDDO
```

← 第1～5桁が“!OCL”
続けて最適化指示子を記述

  - コメント行の形式だが，f90jxコマンドで翻訳するとOCL行として有効化

MPI...ノンブロッキング通信の利用

JAXA Supercomputer System

- MPI_SEND転送処理を，ノンブロッキング通信（MPI_ISEND）に変更することで通信コストを削減

ソース変更前

```
      m = myrank + 1
       if(m.le.npe-npl) then
       mm = m+npl
       call MPI_SEND(Buf1s,jmax1*lmax1*14*2,
& MPI_DOUBLE_PRECISION,mm-1,
& itag1,MPI_COMM_WORLD,IERR)
       end if
       m = myrank + 1
       if(m.ge.1+npl) then
       mm  = m-npl
       call MPI_RECV(Buf1r,jmax1*lmax1*14*2,
& MPI_DOUBLE_PRECISION,mm-1,
& itag1,MPI_COMM_WORLD,status,IERR)
```

ソース変更後

```
      m = myrank + 1
       if(m.le.npe-npl) then
       mm = m+npl
       call MPI_ISEND(Buf1s,jmax1*lmax1*14*2,
& MPI_DOUBLE_PRECISION,mm-1,
& itag1,MPI_COMM_WORLD,ireq1s,IERR)
       end if
       m = myrank + 1
       if(m.ge.1+npl) then
       mm  = m-npl
       call MPI_IRECV(Buf1r,jmax1*lmax1*14*2,
& MPI_DOUBLE_PRECISION,mm-1,
& itag1,MPI_COMM_WORLD,ireq1r,IERR)
       end if
       m = myrank + 1
       if(m.ge.1+npl) then
       call MPI_WAIT(ireq1r,status,IERR)
       end if
       m = myrank + 1
       if(m.le.npe-npl) then
       call MPI_WAIT(ireq1s,status,IERR)
       end if
```

MPI...通信の削減

JAXA Supercomputer System

- 複数の通信メッセージを統合して1回で送受信することで通信オーバヘッドを削減

ソース変更前

```
do l=1,nz
  do j=1,nx
    bufL(j,l)=a(j,1,l)
  end do
end do
call
mpi_isend(bufL,nxz,MPI_REAL4,ilproc,iTag,MPI_COMM_WORLD,iLe,ierr)
 ・・・
do l=1,nz
  do j=1,nx
    bufL(j,l)=b(j,1,l)
  end do
end do
call
mpi_isend(bufL,nxz,MPI_REAL4,ilproc,iTag,MPI_COMM_WORLD,iLe,ierr)
 ・・・
do l=1,nz
  do j=1,nx
    bufL(j,l)=c(j,1,l)
  end do
end do
call
mpi_isend(bufL,nxz,MPI_REAL4,ilproc,iTag,MPI_COMM_WORLD,iLe,ierr)
 ・・・
```

ソース変更後

```
do l=1,nz
  do j=1,nx
    bufL(j,l,1)=a(j,1,l)
    bufL(j,l,2)=b(j,1,l)
    bufL(j,l,3)=c(j,1,l)
  end do
end do
call
mpi_isend(bufL,nxz*3,MPI_REAL4,ilproc,iTag,MPI_COMM_WORLD,iLe,ierr)
 ・・・
```

コードP5の性能チューニング

JAXA Supercomputer System

- FX1基礎性能

非構造格子ソルバー，リストベクトル使用
もともとベクトル用に作成された

① 基礎実行時間

② スレッド間バランス

	経過時間(s)
8プロセス×4スレッド	243.85

	Time(s)	Instructions	MIPS	MFLOPS	L2-miss(%)	Cover(%)
Thread0	244.02	2.41E+11	989.19	262.14	0.755	99.34
Thread1	0.00	6.59E+06	2674.20	154.78	0.113	97.40
Thread2	0.00	6.57E+06	2664.73	152.94	0.060	97.44
Thread3	0.00	6.52E+06	2636.72	145.92	0.058	97.70
全体	246.04	2.41E+11	981.15	259.99	0.755	99.18

③ プロセスあたり性能

ルーチン名	Time(s)	Cost(%)	Instructions	MIPS	MFLOPS	L2-miss(%)	Cover(%)
sub_spdrf1_pri_	81.7	33.19%	2.05E+10	251.0	130.0	4.247	99.6
sub_spdrf1_tet_	42.8	17.41%	8.77E+09	204.7	78.5	5.981	99.6
sub_spvcort_	25.3	10.27%	2.89E+10	1144.8	593.7	0.652	99.6
sub_limiter2_	14.7	5.98%	1.75E+10	1187.9	746.6	0.445	99.6
jwe_gpwd4	8.3	3.36%	1.27E+10	1534.5	677.6	0.008	99.9
isw_check_cqe	5.7	2.33%	1.96E+10	3420.7	5.9	0.001	99.8
strcpy	5.3	2.17%	1.82E+10	3406.1	1.6	0.003	99.7
sub_limiter1_	5.2	2.10%	4.42E+09	856.7	174.4	0.812	99.6
sub_diagonalt_	4.3	1.74%	3.94E+09	921.0	279.4	0.778	99.6
sub_sigmarightt_	3.6	1.48%	1.00E+10	2754.9	578.8	0.047	99.7
プロセス全体	246.0	100.00%	2.41E+11	981.1	260.0	0.755	99.2

コードP5の性能チューニング(2)

- Tune 1.1... 最適化制御行挿入
  - 自動並列化を促進する最適化指示行を挿入

書き換え前(sub_spdrf1_pri.f90)

```
      !cdir nodep
s             DO 100 J=jmin,jmax
p               IE7 = wobj%itcol_pri1(7,J)
p               IE8 = wobj%itcol_pri1(8,J)
p               IE9 = wobj%itcol_pri1(9,J)
p               IE1 = wobj%itcol_pri1(1,J)
p               IE2 = wobj%itcol_pri1(2,J)
p               IE3 = wobj%itcol_pri1(3,J)
p               IE4 = wobj%itcol_pri1(4,J)
p               IE5 = wobj%itcol_pri1(5,J)
p               IE6 = wobj%itcol_pri1(6,J)
                  . . . .
p     100     CONTINUE
```

書き換え後(sub_spdrf1_pri.f90)

```
      !cdir nodep
      !ocl norecurrence  ←最適化指示行
p             DO 100 J=jmin,jmax
p               IE7 = wobj%itcol_pri1(7,J)
p               IE8 = wobj%itcol_pri1(8,J)
p               IE9 = wobj%itcol_pri1(9,J)
p               IE1 = wobj%itcol_pri1(1,J)
p               IE2 = wobj%itcol_pri1(2,J)
p               IE3 = wobj%itcol_pri1(3,J)
p               IE4 = wobj%itcol_pri1(4,J)
p               IE5 = wobj%itcol_pri1(5,J)
p               IE6 = wobj%itcol_pri1(6,J)
                  . . . .
p     100     CONTINUE
```

- Tune 1.2... 配列次元入替え追加
  - 配列q, node, element, edge, triangleについて，C言語のプリプロのマクロを利用し，次元入替えを行いデータアクセスを連続化

マクロ使用ソース

```
#define node(i,j)       Node(j,i)
!cdir nodep
!ocl norecurrence
        DO 1100 J=jmin,jmax
          IEDGE = wobj%itcol_ed1(n3_edge,J)
          I2 = wobj%itcol_ed1(n1_edge,J)
          I3 = wobj%itcol_ed1(n2_edge,J)
           . . .
          X2 = uobj%node(I2,x_node)
          Y2 = uobj%node(I2,y_node)
          Z2 = uobj%node(I2,z_node)
           . . .
 1100   CONTINUE
```

マクロ置換後のイメージ

```
!cdir nodep
!ocl norecurrence
        DO 1100 J=jmin,jmax
          IEDGE = wobj%itcol_ed1(n3_edge,J)
          I2 = wobj%itcol_ed1(n1_edge,J)
          I3 = wobj%itcol_ed1(n2_edge,J)
           . . .
          X2 = uobj%Node(x_node,I2)
          Y2 = uobj%Node(y_node,I2)
          Z2 = uobj%Node(z_node,I2)
           . . .
 1100   CONTINUE
```

コードP5の性能チューニング(3)

JAXA Supercomputer System

- Tune 2.1… 勾配計算における配列定義参照順序の連続化
  - 色分けを削除，要素ごとに計算した勾配の値を一度作業配列に保存し，節点（辺）ごとに和を取るように変更することで，計算の効率化とメモリアクセスの連続

書き換え前

```
      DO 101 ICOLOR=1,wobj%MC_E_N
        . . .
        DO 100 J=jmin,jmax
          . . .
          DRX  = DRX*VO
          DRY  = DRY*VO
          DRZ  = DRZ*VO
          . . .
         wobj%POLE(IP,KRX_PL) = wobj%POLE(IP,KRX_PL) + DRX
         wobj%POLE(IP,KRY_PL) = wobj%POLE(IP,KRY_PL) + DRY
         wobj%POLE(IP,KRZ_PL) = wobj%POLE(IP,KRZ_PL) + DRZ
          . . .
 100    CONTINUE
 101  CONTINUE
!
      DO 101 ICOLOR=1,wobj%MC_PRI_N
        . . .
        DO 100 J=jmin,jmax
          . . .
          DRX  = DRX0*V0 + DRX1*V1 + DRX2*V2
          DRY  = DRY0*V0 + DRY1*V1 + DRY2*V2
          DRZ  = DRZ0*V0 + DRZ1*V1 + DRZ2*V2
          . . .
         wobj%POLE(IP,KRX_PL) = wobj%POLE(IP,KRX_PL) + DRX
         wobj%POLE(IP,KRY_PL) = wobj%POLE(IP,KRY_PL) + DRY
         wobj%POLE(IP,KRZ_PL) = wobj%POLE(IP,KRZ_PL) + DRZ
          . . .
 100    CONTINUE
 101  CONTINUE
```

書き換え後

```
!      DO 101 ICOLOR=1,wobj%MC_E_N
         . . .
!        DO 100 J=jmin,jmax
        DO 100 IELM=1,wobj%n_e_f
         if(uobj%element(IELM,ICL_EL).ge.1)then
           . . .
!          DRX  = DRX*VO
!          DRY  = DRY*VO
!          DRZ  = DRZ*VO
           . . .
!         wobj%POLE(IP,KRX_PL) = wobj%POLE(IP,KRX_PL) + DRX
!         wobj%POLE(IP,KRY_PL) = wobj%POLE(IP,KRY_PL) + DRY
!         wobj%POLE(IP,KRZ_PL) = wobj%POLE(IP,KRZ_PL) + DRZ
           . . .
          elm_wk( 1,IELM) = DRX*VO
          elm_wk( 2,IELM) = DRY*VO
          elm_wk( 3,IELM) = DRZ*VO
           . . .
         end if
 100    CONTINUE
! 101  CONTINUE
!     DO 101 ICOLOR=1,wobj%MC_PRI_N
         . . .
!        DO 100 J=jmin,jmax
        DO 100 IELM=1,wobj%n_prism
         if(wobj%icol_pri(IELM).ge.1)then
           . . .
!          DRX  = DRX0*V0 + DRX1*V1 + DRX2*V2
!          DRY  = DRY0*V0 + DRY1*V1 + DRY2*V2
!          DRZ  = DRZ0*V0 + DRZ1*V1 + DRZ2*V2
           . . .
!         wobj%POLE(IP,KRX_PL) = wobj%POLE(IP,KRX_PL) + DRX
!         wobj%POLE(IP,KRY_PL) = wobj%POLE(IP,KRY_PL) + DRY
!         wobj%POLE(IP,KRZ_PL) = wobj%POLE(IP,KRZ_PL) + DRZ
           . . .
          IELMM = wobj%N_EL_F + IELM
          elm_wk( 1,IELMM)=DRX0*V0+DRX1*V1+DRX2*V2
          elm_wk( 2,IELMM)=DRY0*V0+DRY1*V1+DRY2*V2
          elm_wk( 3,IELMM)=DRZ0*V0+DRZ1*V1+DRZ2*V2
           . . .
         end if
 100    CONTINUE
! 101  CONTINUE
       do ip = 1,wobj%N_P_F
         nelm = wobj%ind_tbl(0,ip)
         do ie = 1,nelm
           ielm = wobj%ind_tbl(ie,ip)
           . . .
           wobj%POLE(IP,KRX_PL)=wobj%POLE(IP,KRX_PL)+elm_wk( 1,ielm)
           wobj%POLE(IP,KRY_PL)=wobj%POLE(IP,KRY_PL)+elm_wk( 2,ielm)
           wobj%POLE(IP,KRZ_PL)=wobj%POLE(IP,KRZ_PL)+elm_wk( 3,ielm)
           . . .
         end do
       end do
```

- Tune 2.2… 流束計算で色分けを削除，ループ分割

コードP5の性能チューニング(4)

JAXA Supercomputer System

- チューニング後の性能

① 実行経過時間

チューニング項目	経過時間(s)	性能比
Asis	243.85	1.00
Tune1	63.64	3.83
Tune1+Tune2	54.43	4.48

② スレッド間バランス

	Time(s)	Instructions	MIPS	MFLOPS	L2-miss(%)	Cover(%)
Thread0	53.43	8.88E+10	1661.64	441.45	0.199	98.11
Thread1	24.65	2.27E+10	921.30	429.69	0.538	98.68
Thread2	24.28	2.25E+10	925.30	433.23	0.531	98.68
Thread3	23.48	2.19E+10	930.66	440.93	0.514	98.69
全体	54.54	1.56E+11	2857.05	1009.45	0.340	98.44

③ 1プロセスあたりのコスト分布

ルーチン名	Time(s)	Cost(%)	Instructions	MIPS	MFLOPS	L2-miss(%)	Cover(%)
sub_limiter2._PRL_1_	3.7	6.77%	1.53E+10	4151.6	2467.9	0.455	98.4
sub_spvcort._PRL_1_	3.2	5.80%	2.69E+10	8511.8	4508.0	0.105	98.6
sub_limiter1._PRL_1_	2.4	4.47%	4.25E+09	1744.1	311.0	1.285	99.5
sub_dr._PRL_2_	2.3	4.15%	8.78E+09	3880.7	645.1	0.621	99.0
sub_diagonalt._PRL_1_	2.1	3.81%	3.27E+09	1573.4	478.0	1.516	99.0
sub_dr._PRL_1_	2.2	3.98%	5.35E+09	2468.6	602.0	1.022	98.6
jwe_gpwd4	6.8	12.40%	1.02E+10	1515.2	672.9	0.017	99.9
sub_sigmarightt._PRL_1_	1.4	2.57%	5.14E+09	3663.1	995.6	0.252	99.4
sub_sigmaleftt._PRL_1_	1.4	2.59%	5.21E+09	3684.4	986.3	0.260	99.4
sub_spdrf1_tet._PRL_1_	1.2	2.23%	5.70E+09	4685.2	3108.7	0.501	96.3
プロセス全体	54.5	100.00%	1.56E+11	2857.1	1009.4	0.340	98.4

性能チューニングに当たっての注意

JAXA Supercomputer System

- 可能なチューニングか
  - 実効性能が低くても，プログラムの性質上，それ以上性能向上が無理な場合もある
- 見通し良く，目処を立てて
  - やみくもにやっても，「労多くして益少なし」になる可能性も
  - ホットスポットを見つける　⇒　劇的な改善
  - チューニングしやすいプログラムの書き方，コンパイラの有効利用
- その他
  - 入出力性能も重要
  - ベンダーの言うことは100%信用するな
  - よくわかっている人を探せ

まとめ

JAXA Supercomputer System

- JAXAにおける航空宇宙のHPCアプリケーションの現状と事例を紹介した.
- FX1クラスタを中核とするJAXAのスパコンシステムの導入経緯，設計思想，構成概要を紹介した.
- アプリケーションの性能チューニングについて，性能情報の取得，評価・分析，チューニング方法を紹介した.
- JAXAシステムのMシステム・チューニングガイドを別途ご参考までに配布します.
- JAXAシステムの情報は，https://www.jss.jaxa.jpで見ることができます．ご質問は，info@jss.jaxa.jpまでお願いします.