

# 大規模長時間実行Gridアプリケーションの 実装と評価

産業技術総合研究所  
グリッド研究センター

武宮 博, 田中 良夫  
中田 秀基, 関口 智嗣

# 大規模長時間実行Gridアプリケーションとは



## ● 大規模長時間実行Gridアプリケーション

- ▶ 大規模: 数千台規模の計算機資源を利用
- ▶ 長時間実行: 数週間～数ヶ月にわたって継続実行
- ▶ Grid: Grid上に散在する複数のクラスタ上で実行

細粒度並列処理を行う複数の実行主体が互いに疎に連携しつつ  
長期間にわたり大規模処理を行う。

Ex.

Multi-disciplinary simulation/multi-scale simulation

## ● 問題点

- ▶ Dedicated executionは現実的でない
- ▶ Grid環境のNetwork/計算機資源は不安定
- ▶ 多数の計算機資源を連携して効率的処理を実現しなければならない

# 要件



## ● 柔軟性

- ▶ 動的割付
  - ◎ 必要に応じて動的に計算機資源上で起動、実行
- ▶ マイグレーション
  - ◎ 動的に利用する計算機資源を変更しつつ実行を継続
- ▶ 計算機資源の追加・変更
  - ◎ 当初想定していなかった計算機資源をターゲットとして追加
  - ◎ 計算機属性(利用バッチキュー、CPU数等)の動的変更

## ● 頑健性:

- ▶ 障害検知
  - ◎ 陽的障害:システムダウン、ネットワーク障害等の検知
  - ◎ 陰的障害:長時間実行待ち等の検知
- ▶ 障害復旧
  - ◎ 発生した障害から自動復旧、計算を継続

## ● 効率性:

- ▶ 計算機資源の効率的管理
  - ◎ 数千台規模の大規模計算機資源を効率的に管理
- ▶ 通信、計算の効率的管理
  - ◎ 個々の細粒度処理を効率的に実行、それらの間の通信を効率的に実行

# 既存プログラミングモデルにおける問題点



## これらの要求を全て満足することは困難

### ► GridRPC

#### ◎ 柔軟性 ☺

- ⊕ Co-allocationの必要なし
- ⊕ 実行対象計算資源の動的変更、計算規模の変更が容易

#### ◎ 頑健性 ☺

- ⊕ ネットワーク/計算資源の障害発生を検知し、appにエラーを返却
- ⊕ システムの一部でエラー発生 → 再実行が可能

#### ◎ 高効率性 ☹

- ⊕ 単一クライアントが多数のサーバプログラムを管理することは困難

### ► Grid-enabled MPI

#### ◎ 柔軟性 ☹

- ⊕ Co-allocationが必要
- ⊕ 静的な実行対象計算資源の利用
- ⊕ 計算規模の変更困難

#### ◎ 頑健性 ☹

- ⊕ システムの一部で障害発生 → システム全体の障害につながる

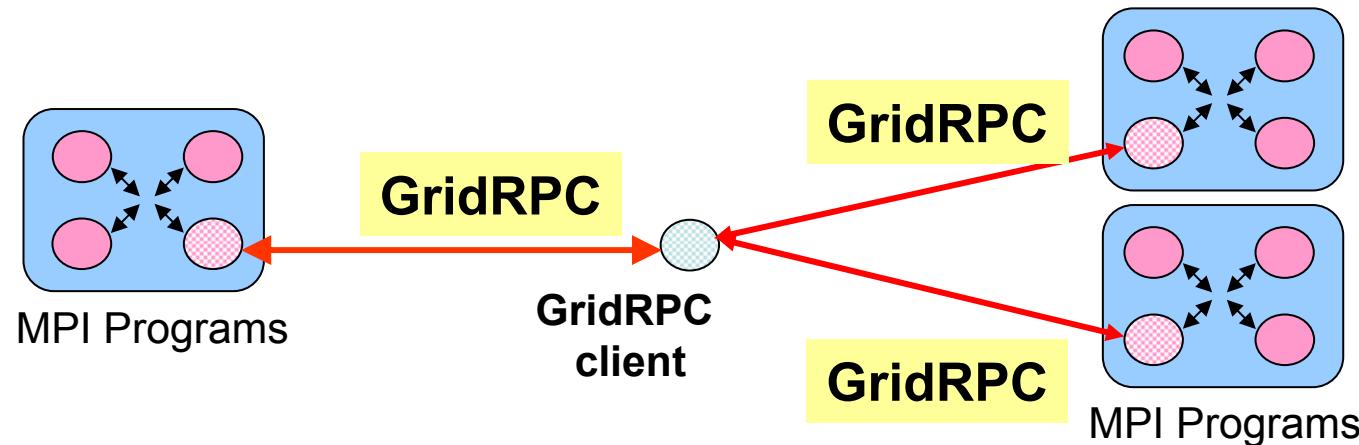
#### ◎ 高効率性 ☺

- ⊕ 効率的な並列実行のための通信機構を提供



## ● GridRPCとMPIを組み合わせて利用

- ▶ MPIにより細粒度並列処理を実現
- ▶ GridRPCによりMPIプログラムを起動、連携
- ▶ GridRPCにより障害を検知



- ▶ 組み合わせることにより両プログラミングモデルの長所を利用

	柔軟性	頑健性	効率性
MPI	-	-	●
GridRPC	●	●	-
MPI + GridRPC	●	●	●

# GridRPC + MPIアプローチの適用



## ● GridRPC + MPIアプローチに基づくGridアプリケーション開発/実行に関する方法論の明確化

- ▶ GridRPC + MPIアプローチに基づく実アプリケーションのGrid化

- © Adaptive Hybrid QM/MD シミュレーションコードを題材として
    - + アプリケーションGrid化の方針
    - + 柔軟性, 頑健性, 効率性の実現

- ▶ 環太平洋Gridテストベッド上で長期継続実行実験を実施

- © 2種類の実行シナリオを想定
    - + 予約ベースの実行
      - 実行前にマニュアルで利用計算機を予約, 予約スケジュールに従い計算機を切り替えつつ実行継続
      - SIMOX Simulation
    - + 遊休計算機の動的利用
      - 事前予約を行わず, その時点で利用可能な計算機を用いて実行継続
      - Friction Simulation

# Adaptive Hybrid QM/MD Simulation



- 高精度大規模材料シミュレーションを現実的な時間内で可能にする

- ▶ 古典MDシミュレーションとQMシミュレーションを連携

- ◉ MDシミュレーション

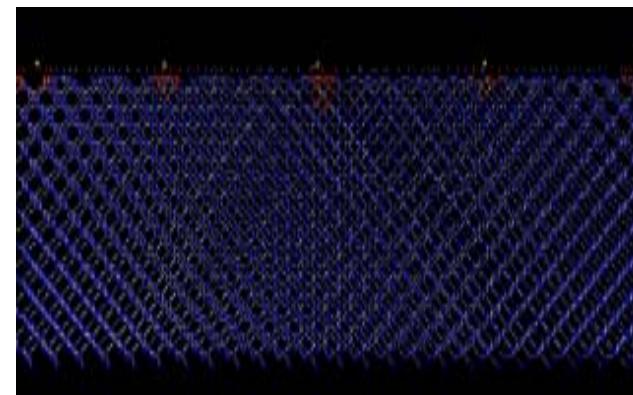
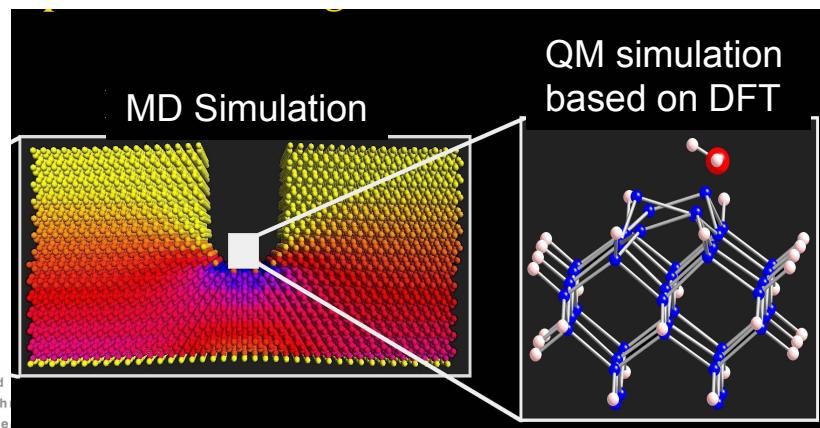
- ⊕ 全領域の原子の振る舞いを計算
    - ⊕ 経験的な原子間ポテンシャルを用いた古典MD

- ◉ QMシミュレーション

- ⊕ 興味のある領域におけるMDシミュレーションの結果を修正
    - ⊕ density functional theory (DFT)に基づく計算

- ▶ 動的に計算領域が変化

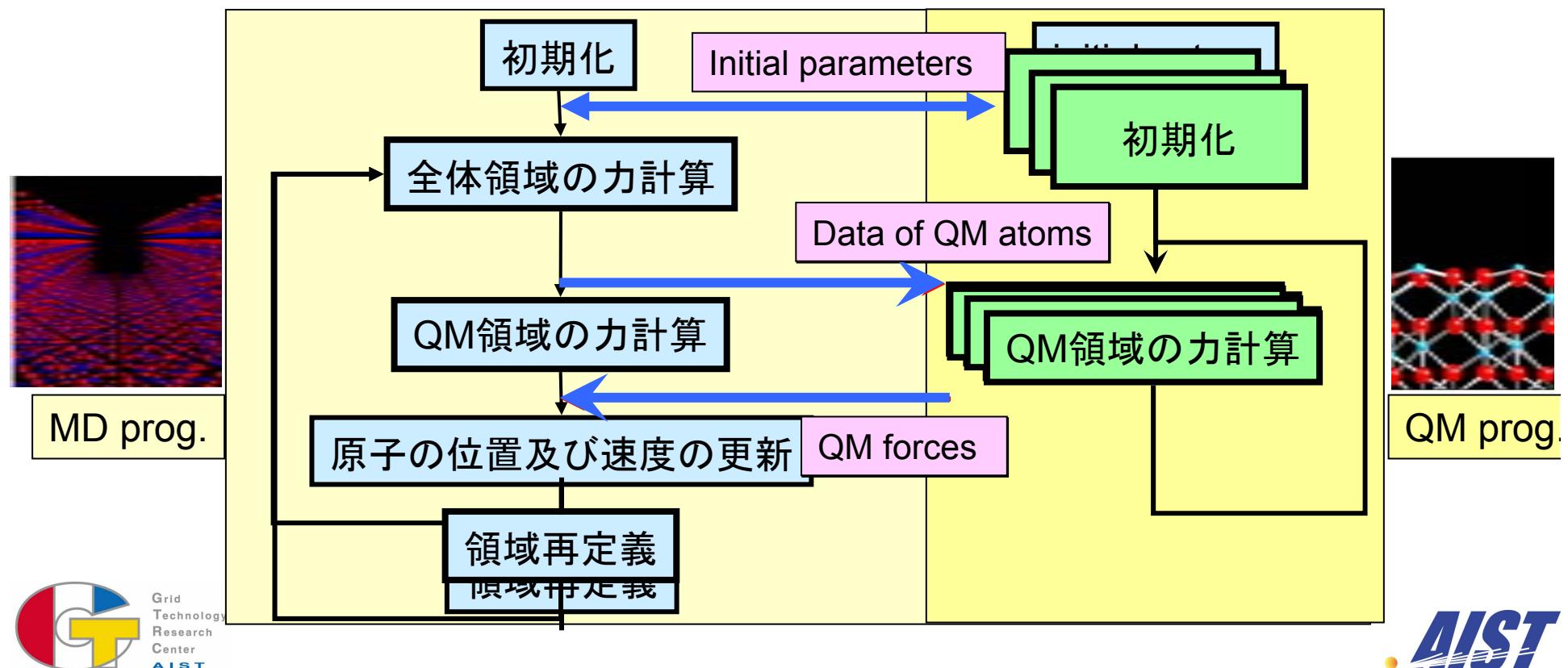
- ◉ 計算精度の保持, 計算量削減



# QM/MDプログラムのGrid化



- 初期化関数の追加
- QMシミュレーション処理を関数化
- QM-MD間通信部分をMPIからNinf-G関数へ変更
- 動的に領域再定義を実行



# Mechanisms for long-run



- アプリレイヤとNinfレイヤ間にスケジューリングレイヤを実装
- 実行対象計算機の変更契機を検知
  - ▶ 計算規模の変更:QM領域数, QM原子数の変更をアプリケーションが通知
  - ▶ 実行時間制限:予約時間終了/継続実行時間制限をチェック
  - ▶ 陽的障害検知:Ninf-Gのエラーを利用
  - ▶ 陰的障害検知:Ninf-Gのタイムアウトを利用
    - ② Job start/finish timeout, session timeout, heart-beat timeout
- サーバプログラムを障害発生前の状態に復旧し, 処理を継続
- 実行対象計算機の決定
  - ▶ 2種類の実行シナリオ, 実行計算機選択手法
    - ② 予約ベース実行:大規模クラスタを優先的に利用, 障害発生時は再試行limitを超えるまでretryを繰り返す
    - ② 遊休クラスタの動的利用:小規模クラスタを優先的に利用, 障害発生時は他のクラスタを利用する
  - ▶ 以下の情報を基に次の計算機資源を決定
    - ② 予約期間(予約ベース実行のばあい)
    - ② 必要とされるCPU数と利用可能なCPU数
    - ② 過去の利用履歴
- 各タイムステップ実行開始時にクラスタ情報設定ファイルを読み込み  
動的にクラスタを追加・削除可能

# SIMOX (Separation by IMplanted OXygen)



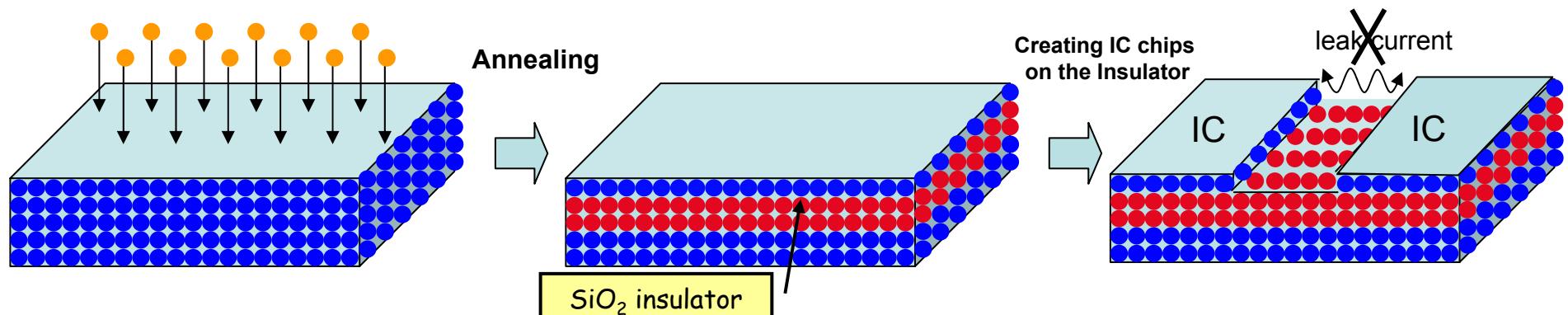
## ● SOI (Silicone On Insulator)構造を持つチップの製造に利用

- ▶ 酸素イオン (~200 KeV)をSi基盤に入射
- ▶ Annealingによる $\text{SiO}_2$ 絶縁層の生成/Si表面の復旧

## ● 高速かつ低消費電力デバイスの製造が可能

- ▶  $\text{SiO}_2$ 絶縁層による漏れ電流のカット

Si基盤に酸素原子を入射



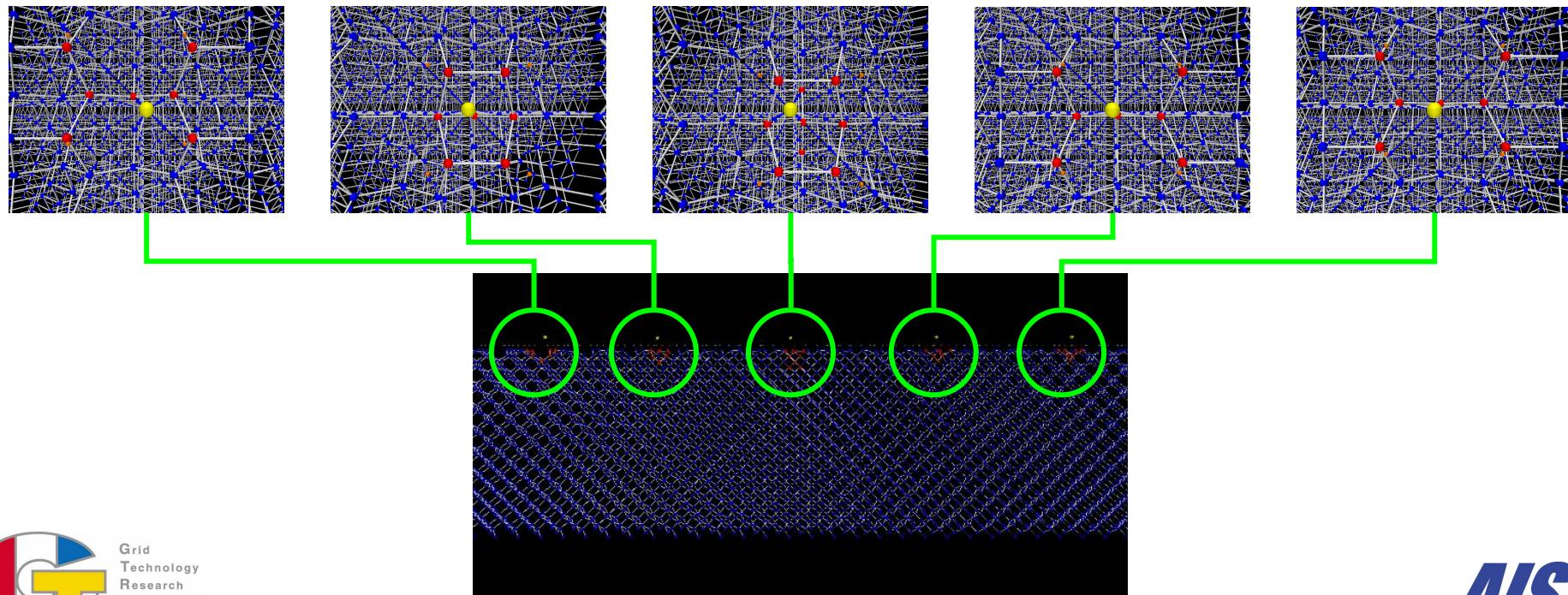
# SIMOX simulation on the Grid



- 予約ベースの実行シナリオに沿って継続実行
- 初期条件

- ▶ Si基盤に5個の酸素原子( $\sim 240\text{eV}$ )を入射
  - ◎ 各粒子はSi結晶の異なる位置に入射
- ▶ 5個のQM領域を定義
  - ◎ 領域数、領域の大きさはシミュレーション中に変化
- ▶ 全原子数: 0.11 million

Initial No. of QM atoms	
Region 0	12
Region 1	12
Region 2	14
Region 3	12
Region 4	12



# SIMOX Simulation用テストベッド

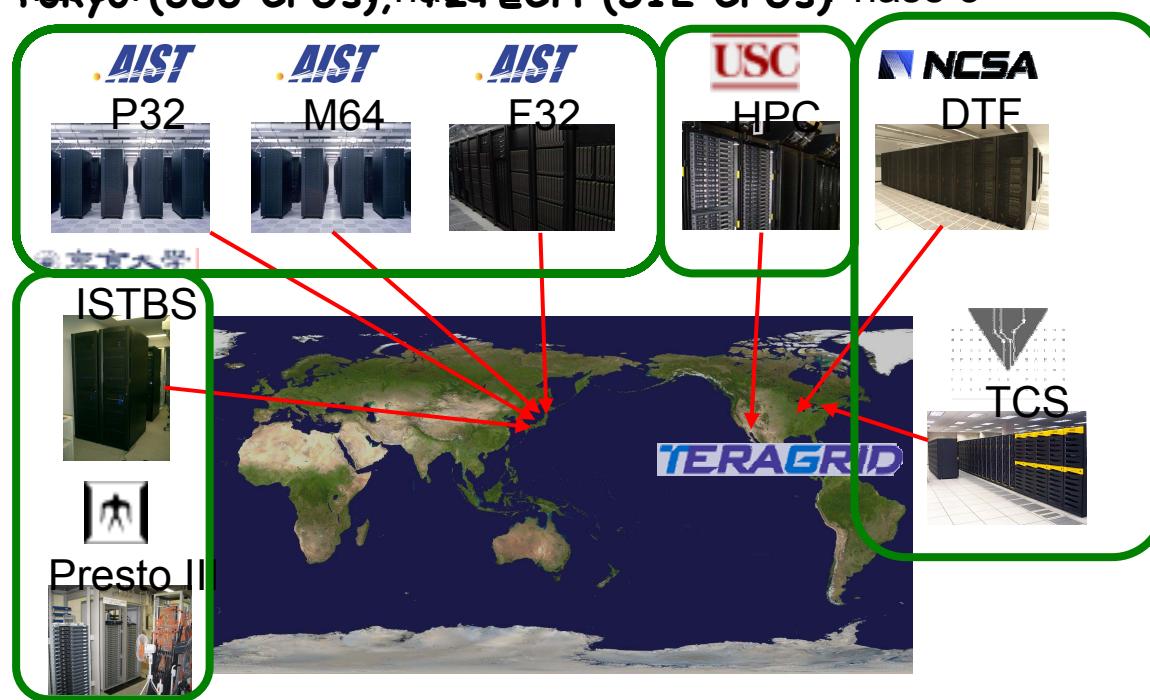


- 8台のクラスタ計16160CPUを用いて実験

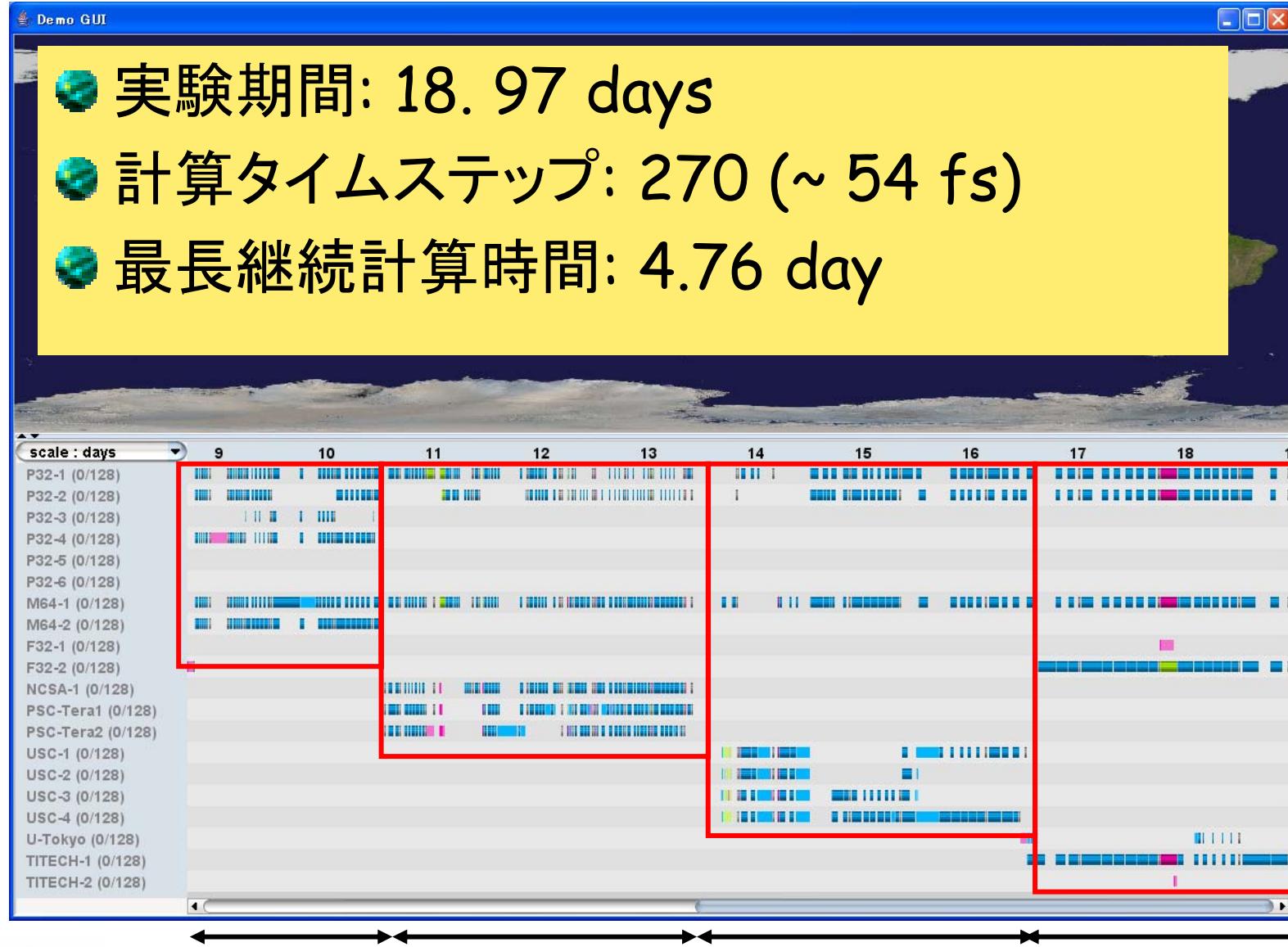
QM1	P32	P32	P32	P32	P32	USC	USC	USC	ISTBS	ISTBS
QM2	P32	P32	NCSA	NCSA	NCSA	USC	USC	USC	Presto	Presto
QM3	M64	M64	M64	M64	M64	M64	M64	M64	M64	M64
QM4	P32	P32	TCS	TCS	TCS	USC	USC	USC	P32	P32
QM5	P32	P32	TCS	TCS	TCS	USC	USC	USC	P32	P32
Reserve	F32	F32	P32	P32	P32	P32	P32	P32	F32	F32

Span Cluster →

@ UPTake1 (386 CPUs) Phase 1 (512 CPUs) Phase 3 Phase 4



# 実験結果



Grid  
Technology  
Research  
Center  
AIST

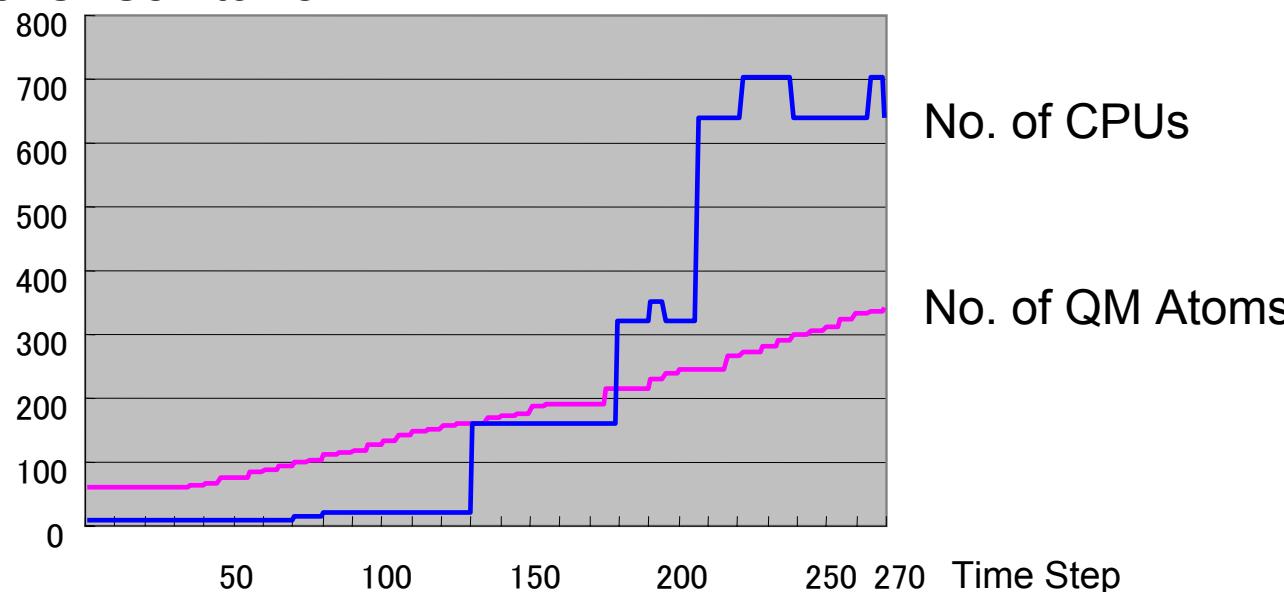


# 柔軟性



- 利用スケジュール、障害発生に応じてクラスタを変更
- QM領域の拡張/分割に対応して利用計算機規模を変更
  - ▶ QM領域の拡張に対応し、MPIプロセス数を増加
    - ◎ QM原子数増加: 47回
    - ◎ QM原子数: 62 → 341
    - ◎ 利用CPU数: 10 → 702
  - ▶ QM領域の分割・統合に対応し、利用クラスタ数を変更
    - ◎ QM領域数の変化: 7回
    - ◎ 状況に応じて領域数が5 or 6に変化

No. of CPUs/Atoms





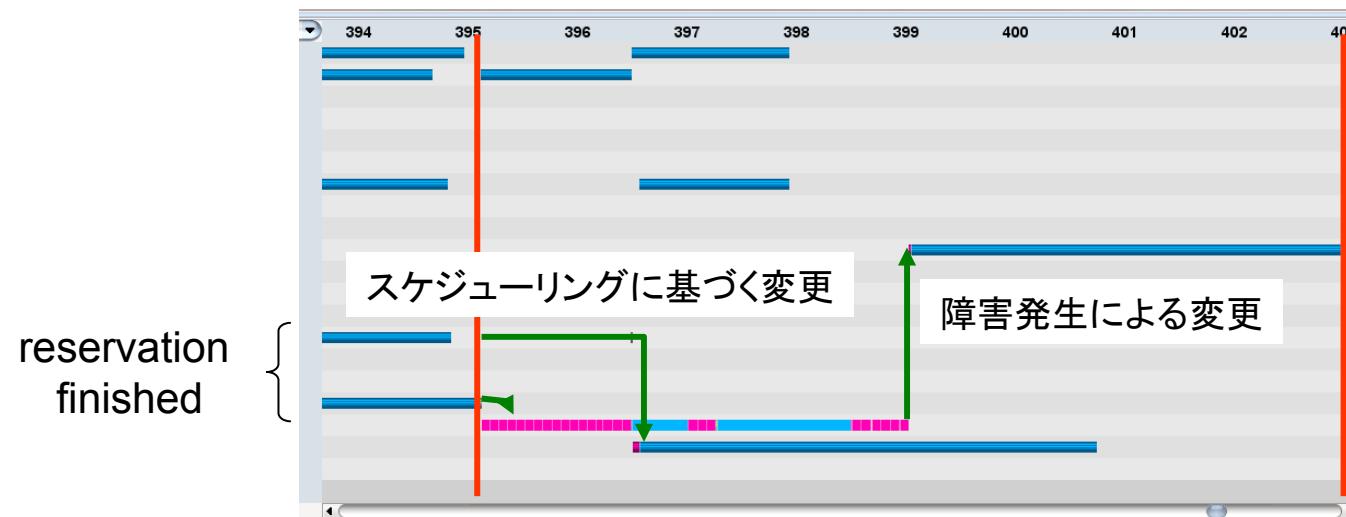
## ● 770回の障害が発生

- ▶ 実行キューが動作しない
- ▶ MPIプログラムの起動失敗
- ▶ quota limit
- ▶ :



22回のマニュアル再起動  
いつまで障害復旧を試みるべきか  
いつから再利用を図ればよいか

## ● 障害検知成功例



# 効率性



## QM-MD間の通信コストはnegligible

- ▶ QM計算時間: ~ 数時間
- ▶ QM-MD間通信コスト: ~ 1分
  - ◎ 転送データサイズ: 1 ~ 5MB/領域

## 実行効率は50~60%程度

- ▶ 平均クラスタ利用率: 2.52 cluster/day
  - ◎ 理想値: 5 cluster/day
- ▶ 主原因:
  - ◎ クラスタ間の負荷のインバランス
  - ◎ 単一クラスタへのQM領域の多重割り付け (shrink mode)
  - ◎ 障害復旧コスト

# Friction Simulation



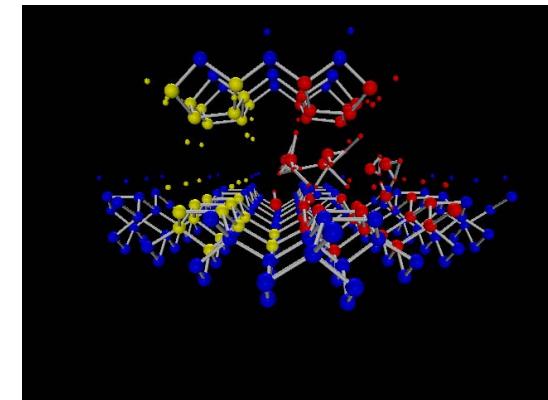
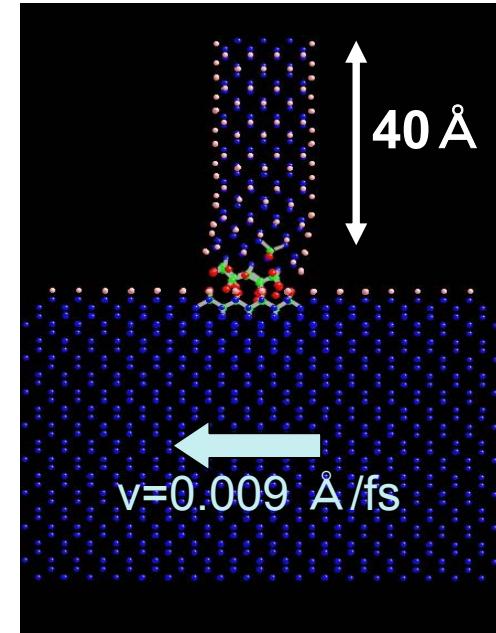
- Nano-scale probing deviceにおける摩擦過程を力学的・化学的に解析

- ▶ Stick-slip過程
- ▶ 接触領域の酸化

- 遊休計算機の動的利用シナリオに沿って継続実行

- 初期パラメータ

- ▶ 基盤移動速度:  $0.009 \text{ \AA/fs}$
- ▶ 2つのQM領域を定義
  - ◎ 72 + 41 QM atoms
- ▶ 総原子数: 28598



# Friction simulation用テストベッド



- 12台のクラスタ計632CPUを用いて実験

- ▶ AIST Super Cluster

- M64 (128 CPUs), F32-1(128 CPUs), F32-2(128 CPUs)

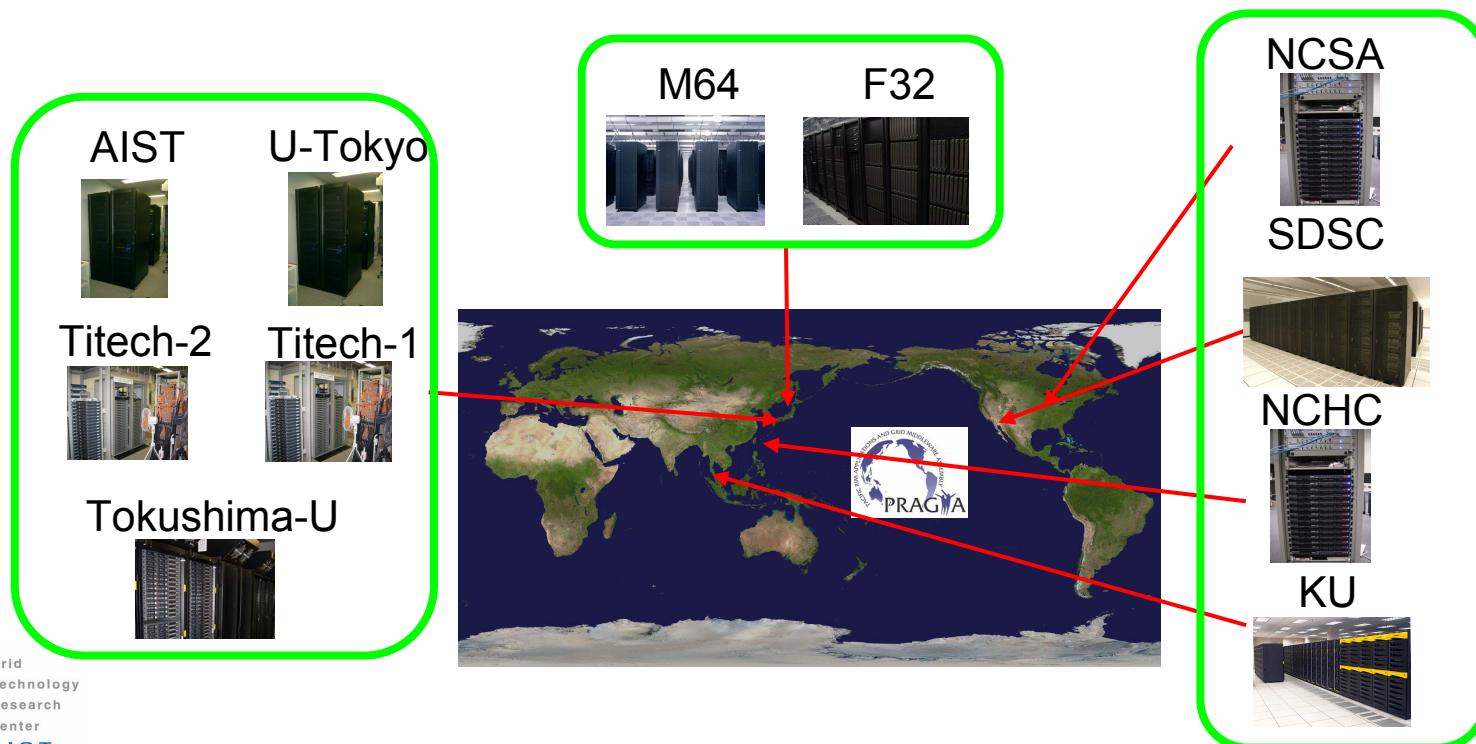
- ▶ Japan Clusters

- U-Tokyo (128 CPUs), Tokushima-U (32 CPUs), Titech-1 (16 CPUs)

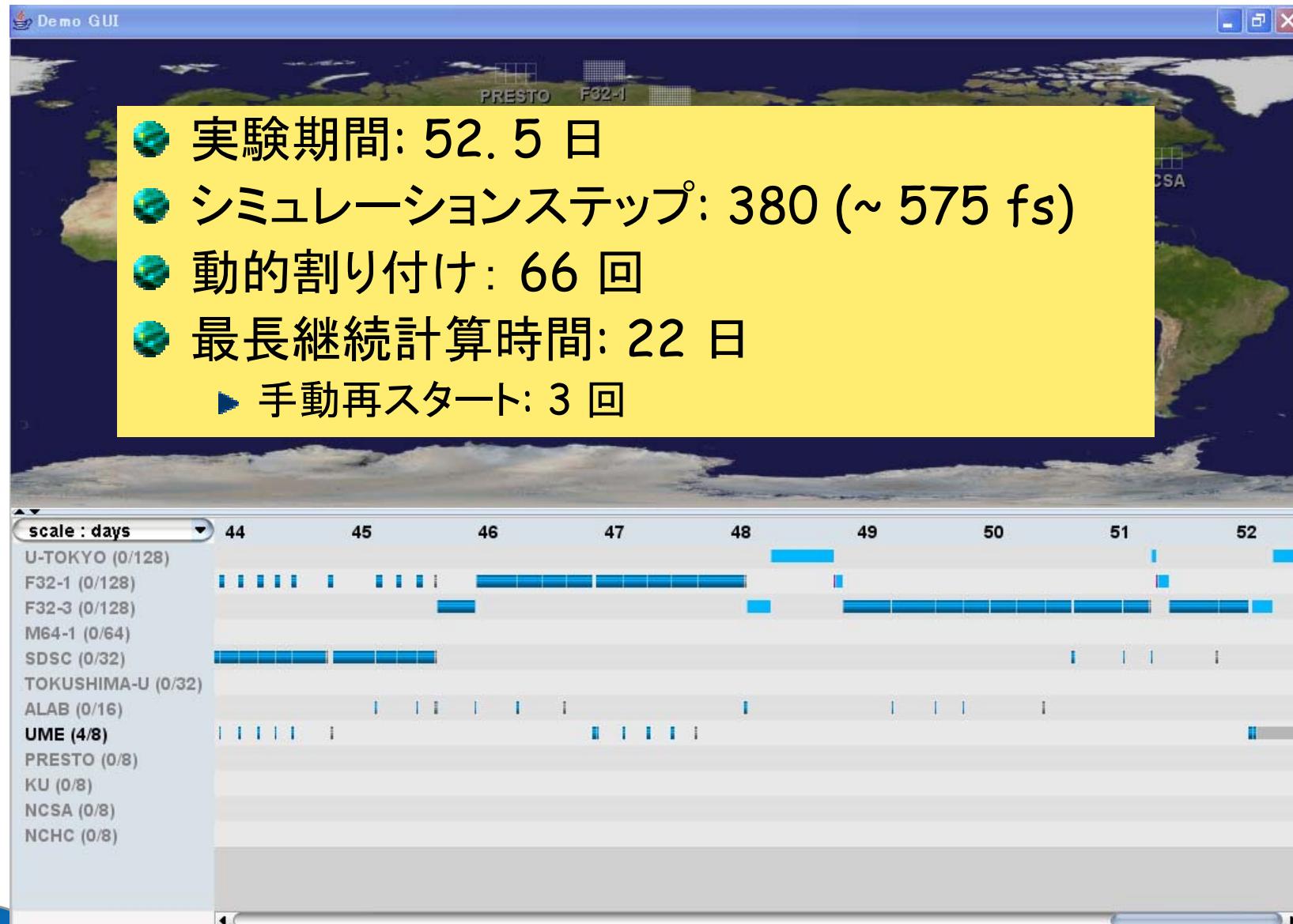
- Titech-2(8 CPUs), AIST(8 CPUs)

- ▶ PRAGMA Clusters

- SDSC (32 CPUs), KU (8 CPUs), NCSA (8 CPUs), NCHC (8 CPUs)



# Result of the Friction Simulation



# Discussion



## ● 大規模プログラムの長期実行に関する困難

### ▶ 運用管理上の問題

#### ② 各サイトのセキュリティポリシに起因する問題

- + Outgoing connectionの制限
  - 内向きだけでなく外向きのconnectionも制限される
- + 細かな非均質性の存在
  - ジョブ実行時間、バッチシステムオプション、ディスクパーティション、quota制限等が異なる

#### ③ 組織間の情報伝達に起因する問題

- + 必要な情報が組織間でうまく伝わらない
- + タイムゾーンが異なる地域間でのネゴは人的負荷が高い
- + Operationの自動化が必要

### ▶ 技術的な問題

#### ④ 幾つかの障害は再実行では復旧不可能

- + Queueが動作しない、quota制限によるファイル出力の失敗...

#### ⑤ Fragileなmiddleware/infra

- + 大規模になればなるほどプログラムの起動に失敗することが多い。
- + mpichはIPCを保持したまま異常終了する

# Summary



- 大規模長時間シミュレーションを想定した2種類の実行シナリオに基づき、環太平洋Gridテストベッド上でQM/MDシミュレーションを実行し、GridRPC+MPIを用いたGrid化手法の有効性を検証
  - ▶ 予約ベースアプローチ
    - ◎ 19日間の実行(最長5日間の継続実行)に成功
  - ▶ アイドルクラスタでの動的実行アプローチ
    - ◎ 50日間の実行(最長21日間の継続実行)に成功
    - ◎ 計算資源が充分にあれば安定して動作
      - ⊕ “君子危うきに近寄らず”戦略が有効
- 実用レベルの長期実行には多数の問題点が存在
  - ▶ 予約ベースの実行のための運用体制の確立が必要
  - ▶ 実行継続をどこまで頑張るか、どこであきらめるか、判断基準は何か、必要情報提供手段はあるか？
  - ▶ 利用CPU数が大規模になればなるほど利用可能クラスタ数が減少
    - ◎ 一つのクラスタの安定性がシミュレーションの継続可能性を左右