

Ninf-G(GridRPC)の利用事例： hybrid QM-CLシミュレーション

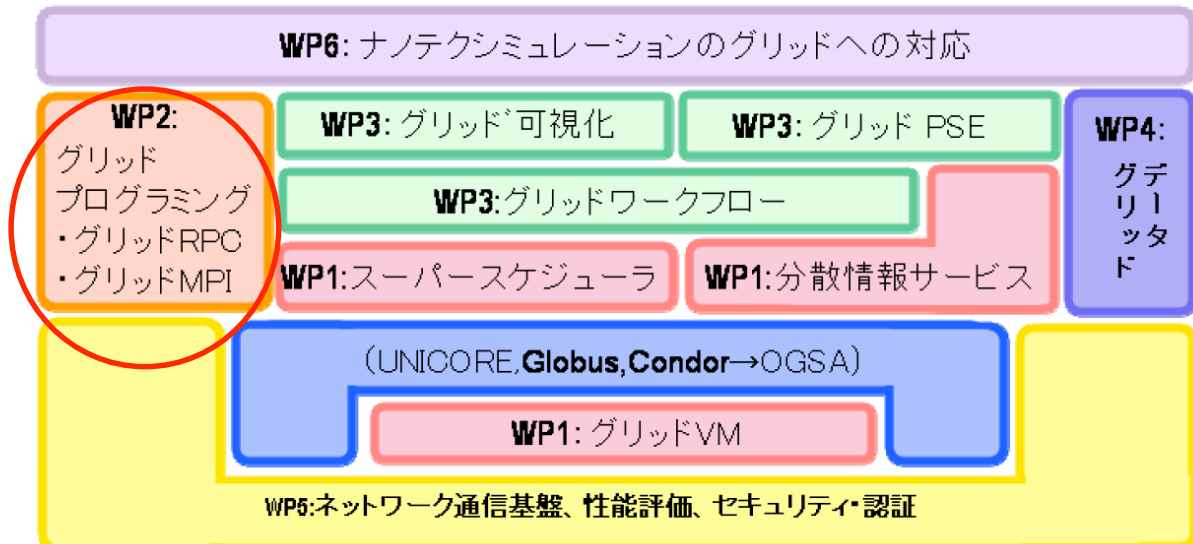
名古屋大学
名工大院
JST-CREST
河野貴久

報告内容

- GridRPCについて
- 利用事例
 - hybrid QM-CL(量子-古典)シミュレーション
- GridRPCの実行事例

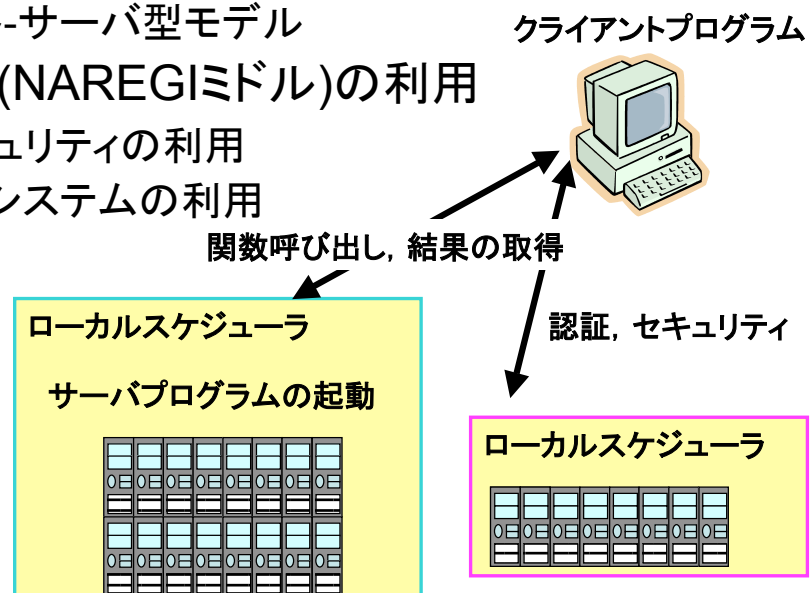
NAREGIミドルウェアのGridRPC

- Ninf-G(産総研グリッド研究センターで開発)



GridRPC(Grid Remote Procedure Call)

- 遠隔関数呼び出し手続き
 - 遠隔地に配置された複数のコンピュータに対して関数の呼び出しを行う
 - クライアント-サーバ型モデル
- グリッド機能(NAREGIミドル)の利用
 - 認証, セキュリティの利用
 - ジョブ起動システムの利用



GridRPCとNinf-G

- GridRPC API
 - OGF(Open Grid Forum)で標準化
 - クライアントAPIが標準化
- Ninf-GはGridRPCの実装の一つ
 - 他のGridRPCの実装は, gridsolve, DIETなど
 - Ninf-GではGridRPC API以外の独自APIも実装 (関数名の最後に_npが付く)

クライアントプログラム

```
grpc_initialize()
grpc_function_handle_init()
grpc_call()
grpc_function_handle_destruct()
grpc_finalize()
```

サーバプログラム

IDL(Interface Definition Language)

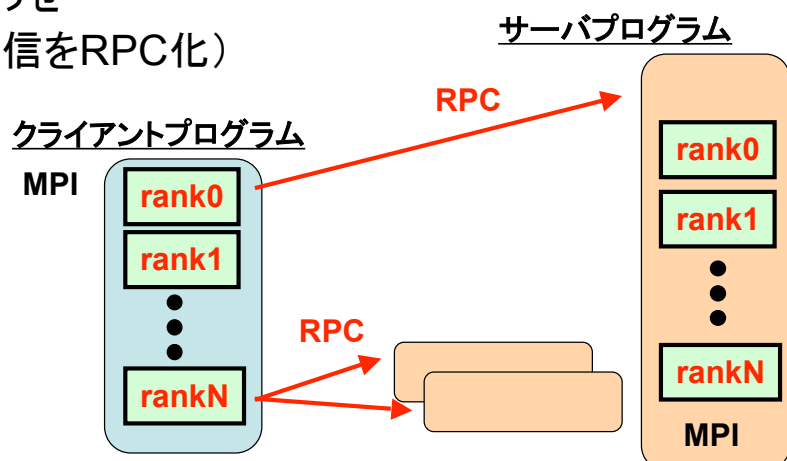
```
Define func( IN int a,
              Out int *b )
{
    *b = calc( a );
}
```

ユーザ関数

```
int calc( int a )
{ }
```

GridRPCのプログラムモデル

- GridRPCを使うことで実装が容易
 - 動的な資源の割り当て, 解放
 - 障害の検知→復旧
- 典型的な利用事例
 - パラメータサーベイ
 - MPIと組み合わせ (グループ間通信をRPC化)



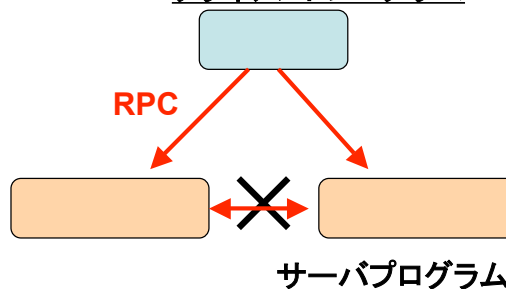
GridRPCを用いた障害検知と復旧

- 障害検知と復旧を組み込んだプログラムの一部
 - GridRPC APIの戻り値を調べ、その後の対応を記述する必要がある

```
count = 0;
for ( i=0; i<30; i++ ) {
    res = grpc_call( &handle, &count );
    if ( res != 0 ) {
        grpc_function_handle_destruct( &handle );
        grpc_function_handle_init( &handle, host_name,
        func_name );
        printf("fail: %d\n", i);
        i--;
        continue;
    }
    printf("success: %d\n", i);
}
```

GridRPCを利用する場合

- MPIの代わりに使うのは難しい
 - 関数呼び出しの時間コストが大きい
 - 通信回数の少ないアプリケーション利用向け
 - サーバ間で直接通信ができない
 - クライアント-サーバ型、関数呼び出しなのでできない
 - クライアント経由で通信を行う必要がある
- クライアントプログラムは障害対策がない
 - バッチ処理で対処すればよい クライアントプログラム



GridRPCのまとめ

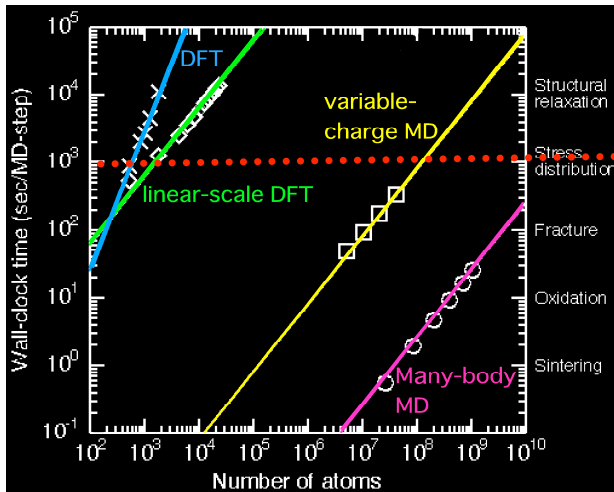
- 遠隔地にある計算機上で関数の実行が可能である
- クライアント–サーバである
- 障害検知と復旧を実装したアプリケーションの開発が可能である
- 実行中に計算機資源を変更可能である
- ユーザがプログラムを書く必要がある

報告内容

- GridRPCについて
- 利用事例
 - hybrid QM-CL(量子-古典)シミュレーション
- GridRPCの実行事例

シミュレーションの計算時間

- 分子動力学(MD)法で必要となる主な計算
 - 古典(CL)的計算による(経験的)相互作用
 - 計算量は少ないが, 物理現象の精度は低い
 - 量子(QM)的計算による(密度汎関数法,DFT)相互作用
 - 計算量は多いが, 物理現象の精度は高い

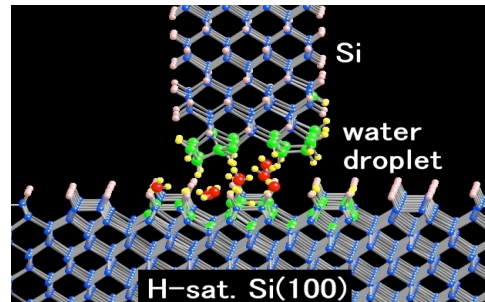
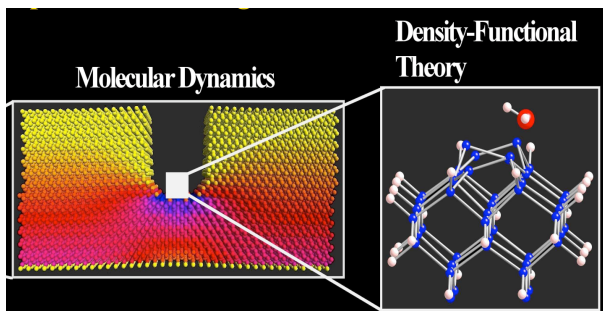


現実的には計算時間は
1000秒以下が望ましい
精度の高いDFT計算では
数百原子程度の規模が限界

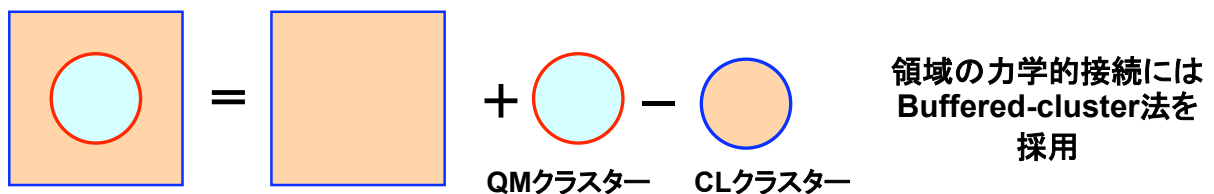
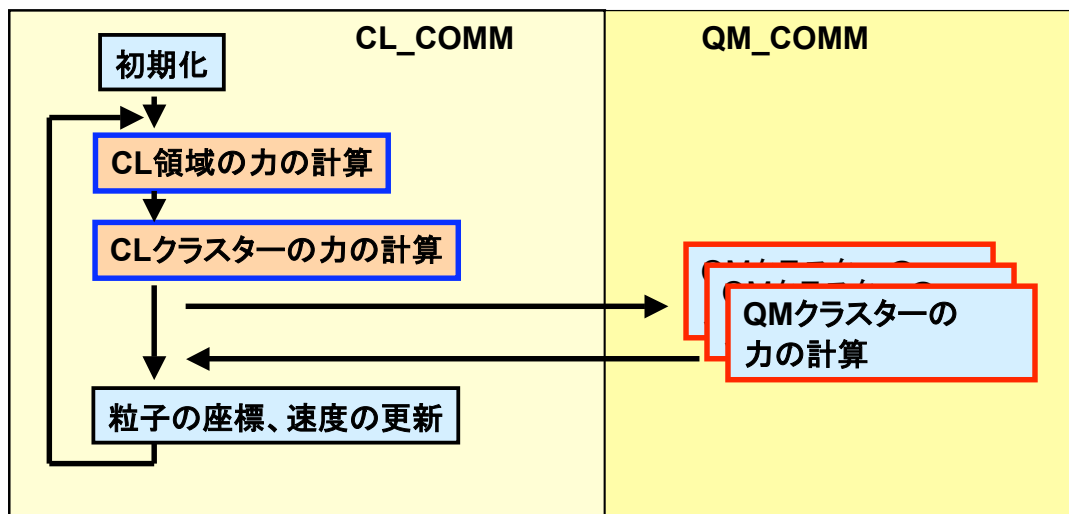
Benchmark tests
on 1024-node Cray T3E (2000)

hybrid QM-CL法

- 注目する領域に精度の高い量子計算領域を適用し周辺領域には古典計算領域を用いて境界条件と扱うことで, 精度の高い大規模系のシミュレーションを行う方法
- 物理現象に対して, 適切な精度の計算を適用できる
- 高い精度の量子計算領域を大きくとる必要がないため計算コストを少なくできる
- 量子領域の動的な領域変更に対応

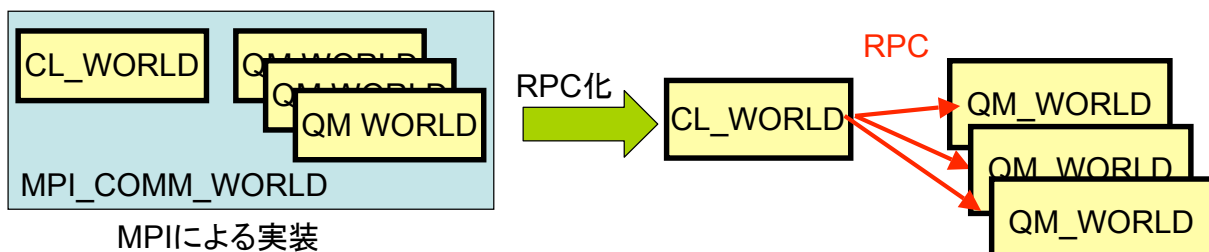


hybrid QM-CL法の計算プロセス



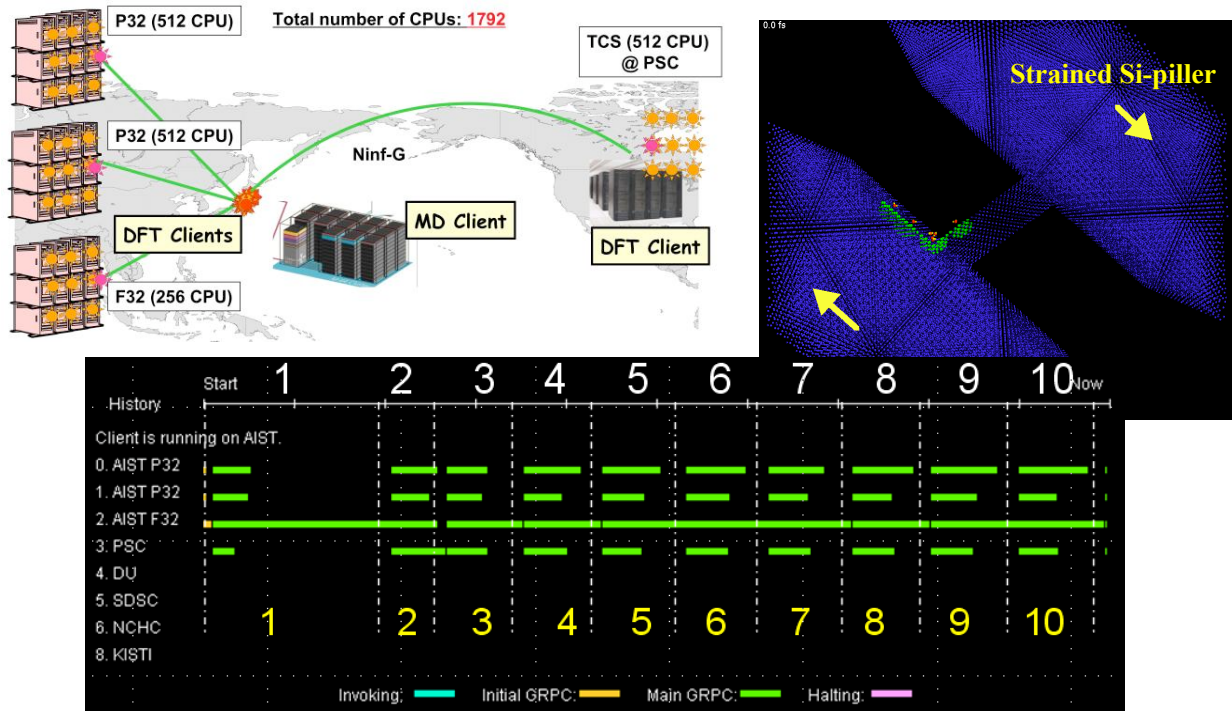
hybrid QM-CL法のGridRPC化

- QM計算を関数化
- MPIのグループ間通信の代わりにRPCを採用
- QM計算に関しては, 計算資源を動的変更
- QM計算に関しては, 障害による停止からの再実行



- 計算時間
 - MD計算時間: 数秒/step
 - QM計算時間: 数十分/step
 - QM-CL間通信データ量: 数秒 (数十kB程度)

実行デモ: SC2004

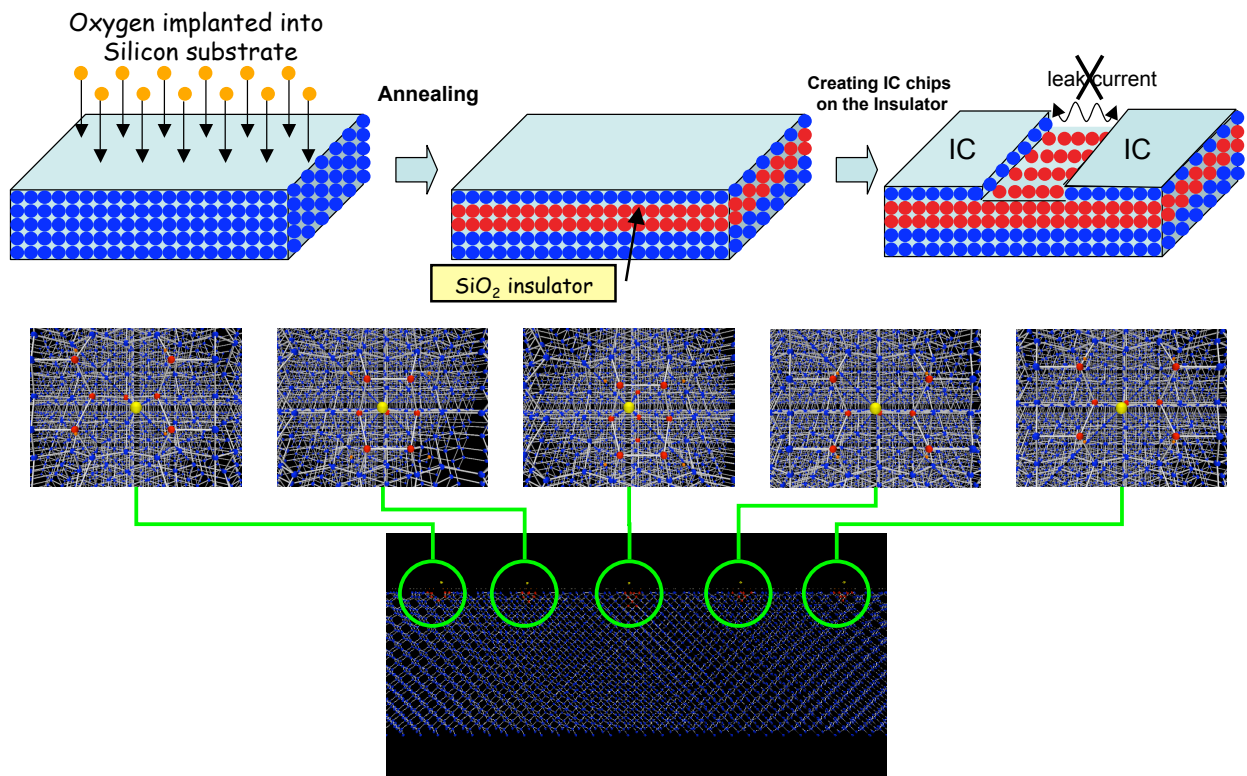


- 10 hours of continuous run on 1792CPU
- Nearly perfect Grid parallelization

Collaboration: Dr. Tanaka (Grid Center, AIST) 2004

実行デモ: SC2005

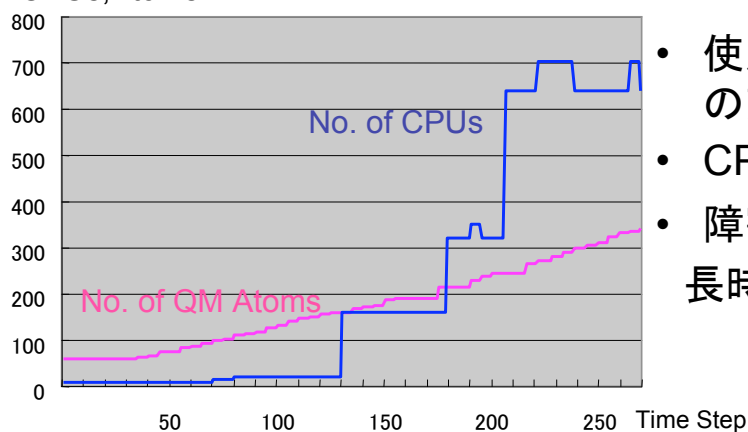
- SIMOXシミュレーション



実行デモ: SC2005

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

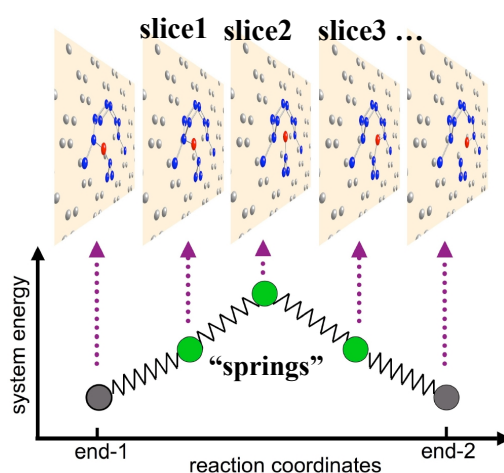
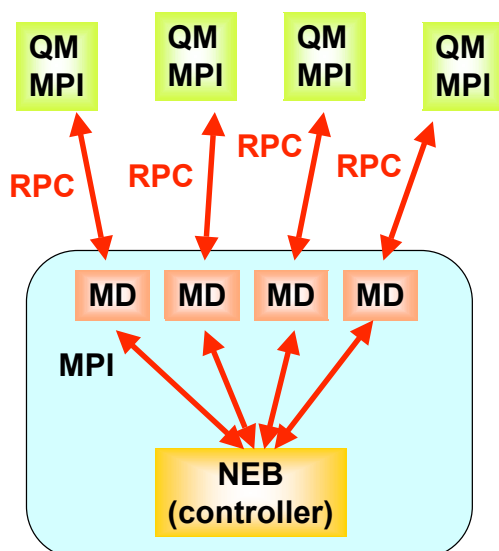
No. of CPUs, Atoms



- 使用する計算機サイトの変更
- CPU数の動的変更
- 障害検知による長時間シミュレーション

実行デモ: SC2007

- NEB(Nudged Elastic Band)法+ハイブリッドQM-CL法
- g-Al₂O₃中のH原子の拡散
- 経路とバリアエネルギーの計算
- 経路上の30slices



Force on atom-*i*:

$$\vec{F}_i = -\nabla V(\{\vec{R}_i\})_{\perp} + \vec{F}_i^{\text{spring}}_{\parallel}$$

tangential to slice sequence

parallel to slice sequence

hybrid QM-CLシミュレーションのまとめ

- QM計算は, QM-CL間の通信時間に対して大きな計算時間を必要とし, また関数として扱うことができるためグリッドに適用しやすかった
- GridRPCを用いることで計算資源の動的割り当てが可能となった
- GridRPCによる障害対策をすることで長時間のシミュレーションが可能となった