

自動計算システム

仮定

- 計算サーバーはPCと別に存在する。
- 計算サーバーが計算している状態はPCからは分からない。計算サーバーはbatchを用いて計算を行う。

PCでのstatusと計算サーバーでのstatusの2つの状態を持つ。

自動計算システム

Inputdata/fcc/random_structure1

Inputdata/fcc/random_structure2

Inputdata/fcc/random_structure3

...

Inputdata/fcc/deformedlattice_structure1

Inputdata/fcc/deformedlattice_structure2

Inputdata/fcc/deformedlattice_structure3

...

Inputdata/hcp/**deformedlattice_structure1**

ID1.0

ID2.0

ID3.0

ID4.0

ID5.0

ID6.0

...

Autcc (半自動計算システム)

はじめに人間が行うこと。
directory毎のデータ作成
.IDの作成(hash.pyによる)
DB内はIDにより参照される。

DB作成
\$prog -createdb

DBへの(再)登録
\$prog -init

計算パラメタの追加
計算サーバーへ送る
ファイルの作成
\$prog -send

計算終了判定
\$prog -recv

サーバーから
持ってきたデー
タの状態の取り
込み
\$prog
-updatestatus

計算収束

ServerA
Submit jobs

ServerB
Submit jobs

ServerC
Submit jobs

誤って\$progを実行しても
データは壊れないように
作られている。

autccの実装

- autcc -createdb

databaseを作る。

- autcc -init

Inputdir/以下でfile .IDがあるdirectoryを検索する。

IDがDBにある場合はstatus=(new,idle)にする。count=0

IDがDBにあるものに対しては何もしない。(inputdir/にdirectoryを随時追加して良い。)

- autcc -send

calcddata/ID.count/をつくる。countによりinputfileをつくる。入力ファイルからinputfileをつくるroutineをつくる必要がある。

(count>0の場合は先の計算で失敗しているので収束パラメタを変更することになる。)

必要なファイルのコピーなども可能。

calcddata/ID.count/.EXECSTATUSをつくる。サーバー上の計算状態を表す。ファイルの中身はidle。

IDの状態を(submitted,idle)にする。

ファイルを計算サーバーへ持っていく。jobが正常終了したらID.count/.EXECSTATUSの中身をfinishedとしておく。(PCからはjobの状態は分からない。)

計算サーバーからファイルをcalcddata/ID.countへコピーする。

- autcc -updatestatus

ID.count/.EXECSTATUSの状態をDBに反映させる。計算が終了していなければ(submitted,idle)、終了していれば(submitted,finished)になる。

autcc -recv

(submitted,finished)のみ処理を行う。

Calcddata/ID.count/output_scf.txt からSCFが収束したかを判定する。収束していたら(calculated,finished)、収束していなかったら(new,idle),count++をする。