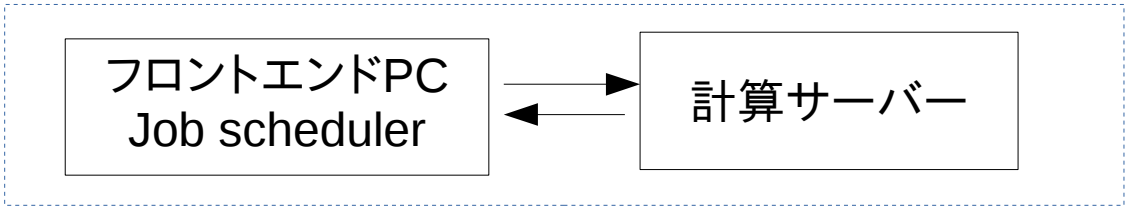
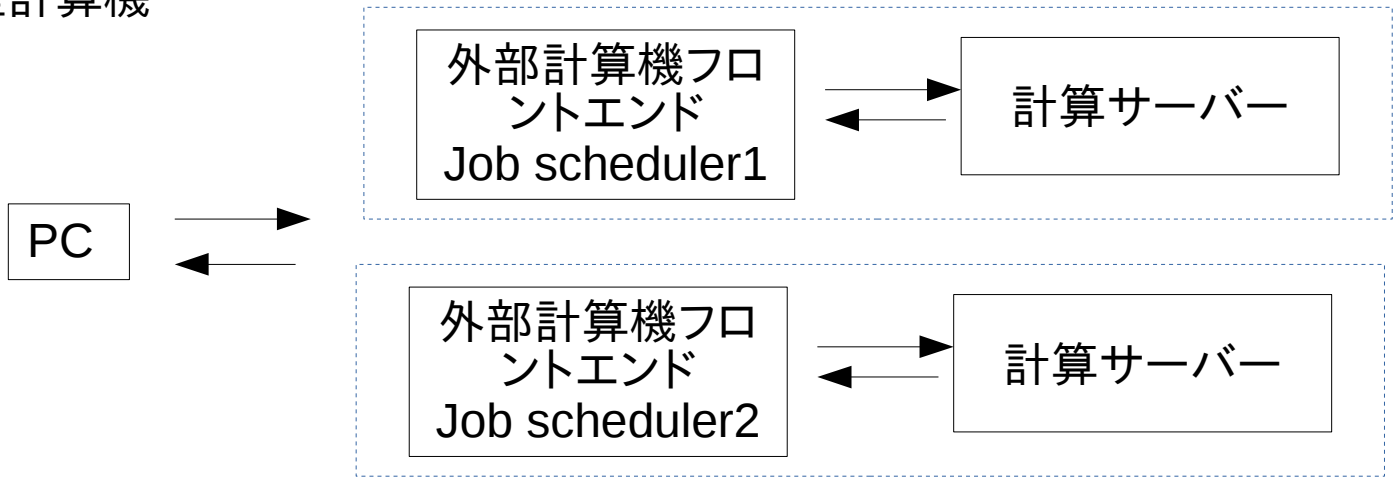
自動計算システム

PCクラスタ



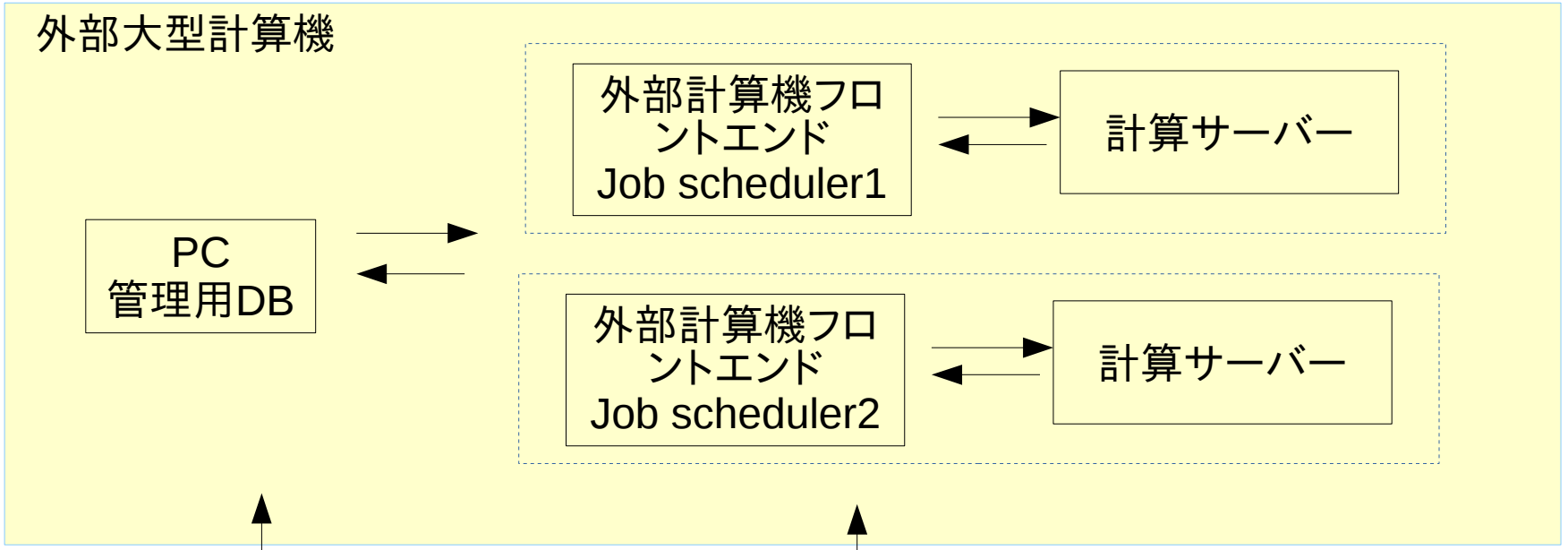
- Job schedulerが動き、jobを管理できる。
- 計算時間は無制限。
- homogeneous環境

外部大型計算機



- PCではJob schedulerが動かないのでjobを管理できない。
- 計算時間は制限される。
- heterogeneous環境

自動計算システム



policy

外部計算機のheterogenousなjob schedulerの面倒は見ない。

- なるべく簡単なjob構成にして外部計算機に持っていけるようにdirectory/fileを作る。
- 終了状態に応じて再計算のためのdirectory/fileを容易に理解できる形で作る。

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。

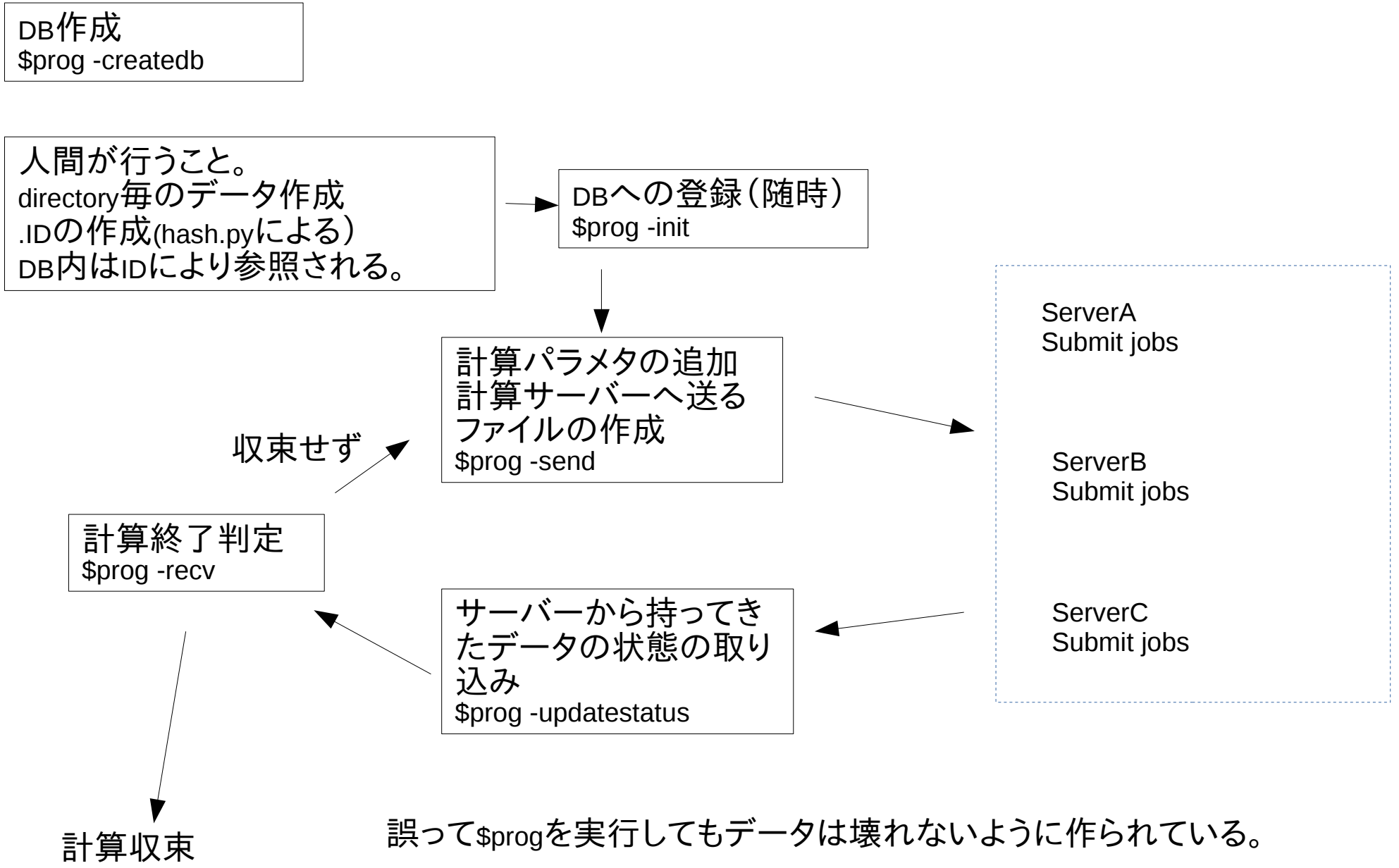
例:

jobが正常終了したこと、programが正常終了したこと、電子状態が収束したこと、はすべて状態として異なる。

jobが正常終了したかどうかは自動計算システムは感知しない。

PCでのstatusと計算サーバーでのstatusの2つの状態を持つ。

Autcc (半自動計算システム)



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。

directory構成

inputdata/	入力データ、結果ファイルが置かれる。
calcdata/	計算サーバーへ送るために一時ファイルが置かれる。 計算終了後は消して良い。

一次元に直す。

Inputdata/fcc/random_structure1

Inputdata/fcc/random_structure2

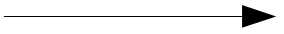
Inputdata/fcc/random_structure3

...

Inputdata/fcc/deformedlattice_structure1

Inputdata/fcc/deformedlattice_structure2

Inputdata/fcc/deformedlattice_structure3



ID1.0
ID2.0
ID3.0
ID4.0
ID5.0
ID6.0
...

…

Inputdata/hcp/**deformedlattice_structure1**

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

calcdata/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。

calcdata/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calcdata/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。

Calcdata/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(claculated,finished)、収束していなかったら(new,idle),count++をする。

autcc利用法

At local PC
$ db11.py -createdb
$ db11.py -init
$ db11.py -send
$ rsync-send-data.sh

At computational server
$ for n in calcdatadirectory; do submit batchfile.sh; done
…
$ for n in calcdatadirectory; do { echo finished > $n/.EXECSTATUS, if job is successfully finished; done}

At local PC
$ rsync-receve-data.sh
$ db11.py -updatestatus
$ db11.py -recv
$ db11.py -purge
If all the jobs are not converged
$ db11.py -send
...