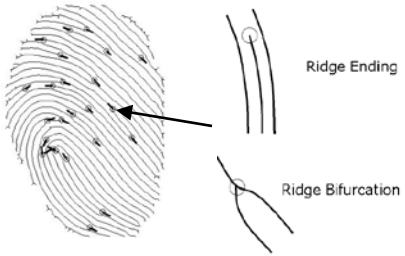
CSE633 Parallel Algorithms
Project Final Report

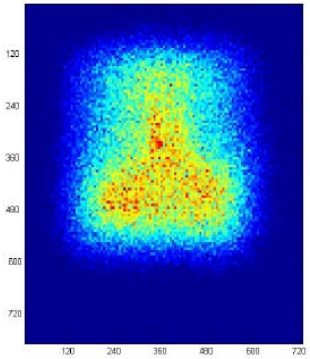
Chang Su
Person #: 34046248
April 27, 2009

Generative Models of Individuality for Fingerprints

Training data set

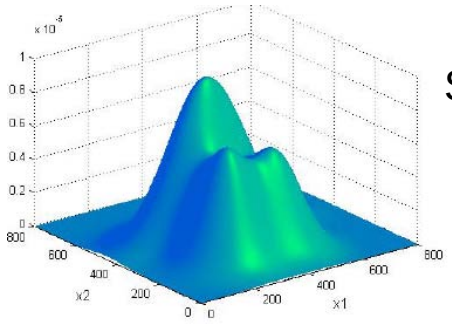


Minutiae x



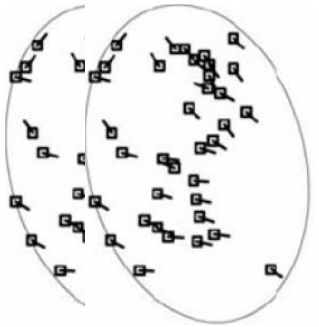
Training

Generative Model p(x)



EM Algorithm

Sampling



Probability of Random Correspondence (PRC)

Sequential EM

Input : N training samples

Output: parameters of each of the K components

1. Initialize the distribution parameters
2. Repeat until *convergence:*
    1. E-Step:
       Estimate the expected value of the unknown variables, given the current parameter estimate.
    2. M-Step:
       re-estimate the parameters to maximize the likelihood of the data, given the estimates of the expectations of the unknown variables.

Basic Ideal of Parallel EM

- E step dominates the execution time on each iteration,
- E step is inherently data parallel if the parameters are available for each processor.
- The loop is parallelized by evenly distributing the data across the processors
- Single Program Multiple Data (SPMD)
- Processor P_0 distributes the data by using MPI_Scatter.
- Call MPI_Allreduce to sum up the local parameters to obtain the new global parameters.
- Only one collective communication is needed in each iteration .

Parameter Estimation using EM

- Minutiae distribution (Mixture of Gaussian and Von-mises)

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(s|\mu_k, \Sigma_k) \cdot \mathcal{V}(\theta|\nu_k, \kappa_k)$$

EM for Mixture of Gaussian and Von-mises

Repeat until *convergence:*

E step: $\gamma_{dk} = \frac{\pi_k \mathcal{N}(\mathbf{x}_d|\mu_k, \sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_d|\mu_k, \sigma_k)}$

M step:

$$\pi_k^{(n+1)} = \frac{1}{D} \sum_{d=1}^{D} \gamma_{dk}^{(n)}$$

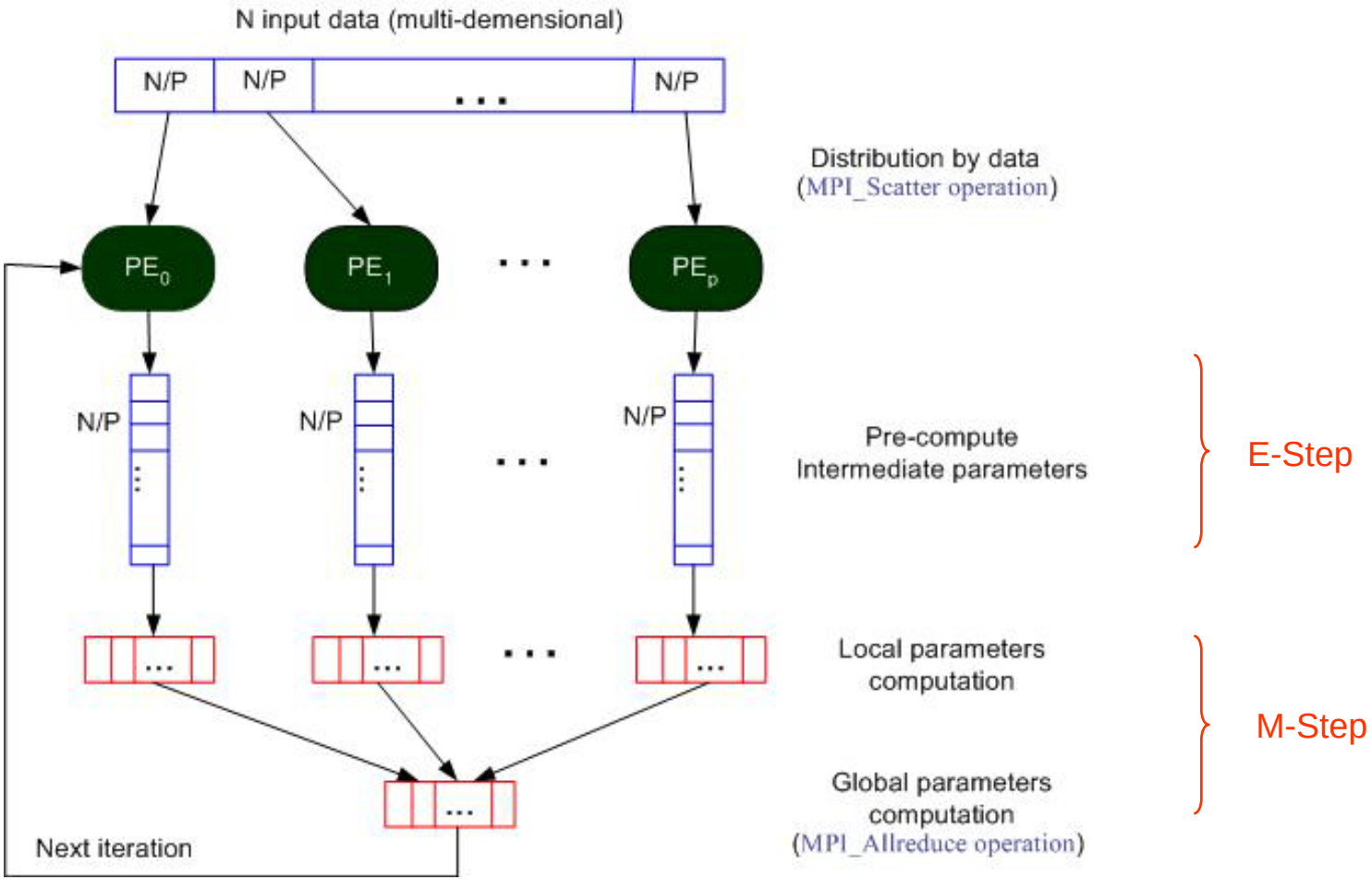
$$\nu_{mk}^{(n+1)} = \frac{1}{2} tan^{-1} \left(\frac{\sum_{d=1}^{D} \gamma_{dk}^{(n)} sin2\psi_d}{\sum_{d=1}^{D} \gamma_{dk}^{(n)} cos2\psi_d} \right)$$

$$\mu_{mk}^{(n+1)} = \frac{\sum_{d=1}^{D} \gamma_{dk}^{(n)} s_m}{\sum_{d=1}^{D} \gamma_{dk}^{(n)}}$$

$$\frac{I_0'(\kappa_{mk}^{(n+1)})}{I_0(\kappa_{mk}^{(n+1)})} = \frac{\sum_{d=1}^{D} r_{dk}^{(n)} cos2(\psi_d - \nu_k^{(n+1)})}{\sum_{d=1}^{D} r_{dk}^{(n)}}$$

$$\Sigma_{mk}^{(n+1)} = \frac{\sum_{d=1}^{D} \gamma_{dk}^{(n)} (s_m - \mu_{mk}^{(n+1)})(s_m - \mu_{mk}^{(n+1)})^T}{\sum_{d=1}^{D} \gamma_{dk}^{(n)}}$$

Parallel EM


N input data (multi-demensional)
N/P
N/P
N/P
Distribution by data
(MPI_Scatter operation)
PE0
PE1
PEp
N/P
N/P
N/P
Pre-compute
Intermediate parameters
E-Step
Local parameters
computation
M-Step
Global parameters
computation
(MPI_Allreduce operation)
Next iteration

Pseudo Codes

```
#include "mpi.h“
……
int main(int argc, char *argv[ ]){
      ……
       double gparam[K] ={…};
       double lparam [K] = {…};

       ierr = MPI_Init(&argc, &argv);
      ……
       if (rank == MASTER){
                  fptr=fopen("minutiae.txt","r“);
                  fscanf(fptr,"%d %d %d",&x,&y,&t);
                  MPI_Scatter(data, numpp, MPI_INT, localData, numpp, MPI_INT, 0, MPI_COMM_WORLD);
       }
       ……
        Iterate until convergence{
                  for (idx=0; idx<nump; idx++)
                              r[idx] =…
         for (idx=0; idx<numpp; idx++)
                              localParam = …
                  MPI_Allreduce(&localParam, &globalParam, K, MPI_DOUBLE, MPI_SUM, MPI_COMM_WORLD);
                  globalParam = …
       }
       ……
       ierr = MPI_Finalize();
       return 0;
}
```

Initialize the global and local parameters

Initialize MPI

Read training data from disk

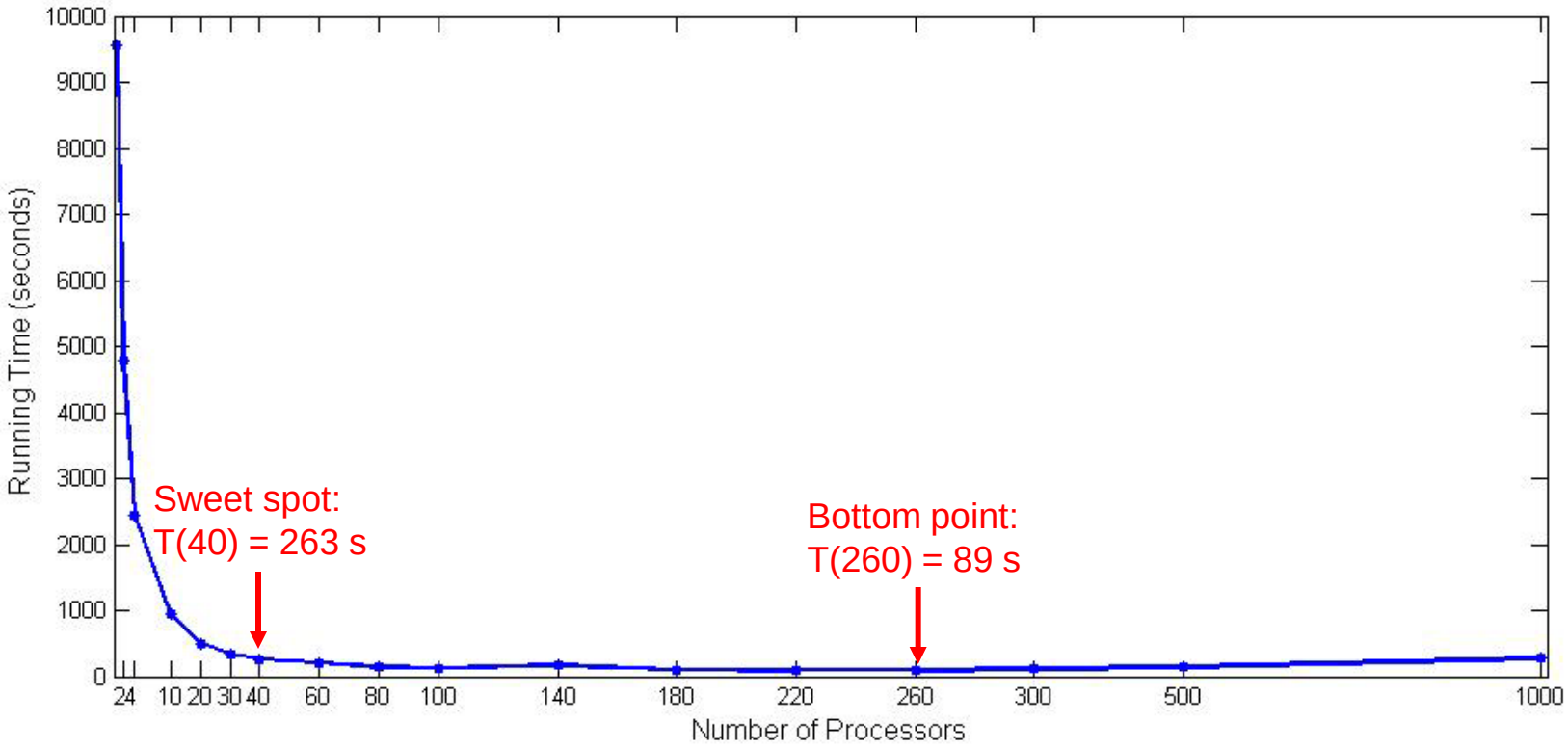
Parallel codes

E-Step

Sequential codes

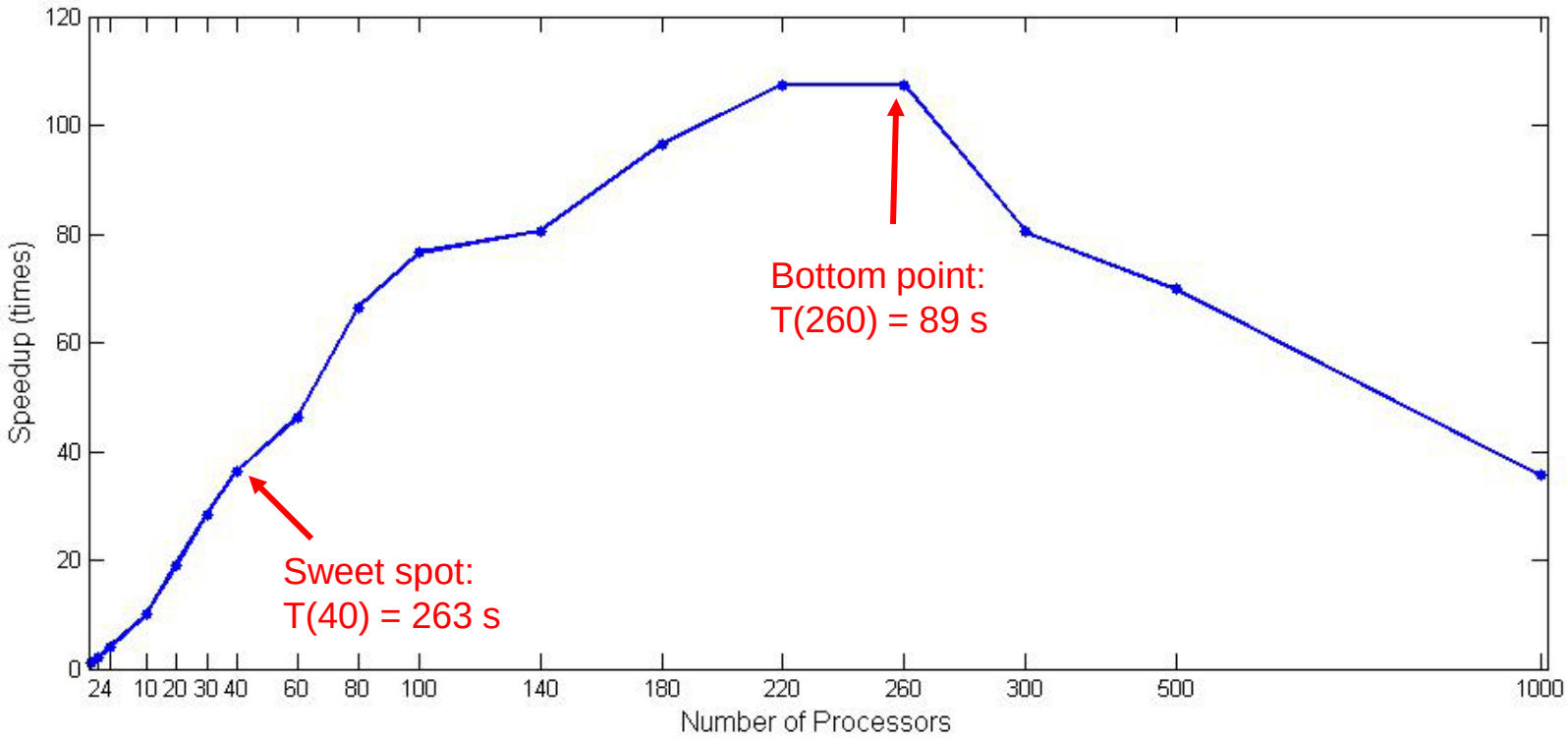
M-Step

Running time with varying numbers of processors



Input data set: 180,000 minutiae from NIST 4

Speedup with varying numbers of processors



Input data set: 180,000 minutiae from NIST 4

Running Time Analysis

Sequential algorithm (one processor):
parallel part: 9556 s >> sequential part: 6 s

Why the best running time is T(260) = 89 ?

- T(s): running time of sequential codes
- $T(c_i)$: parallelization costs with i processors (initialization, communication, etc.)
- $T(p_i)$: running time of parallelized codes with i processors

Running time with n processors: $T(n) = T(s) + T(c_n) + T(p_n)$

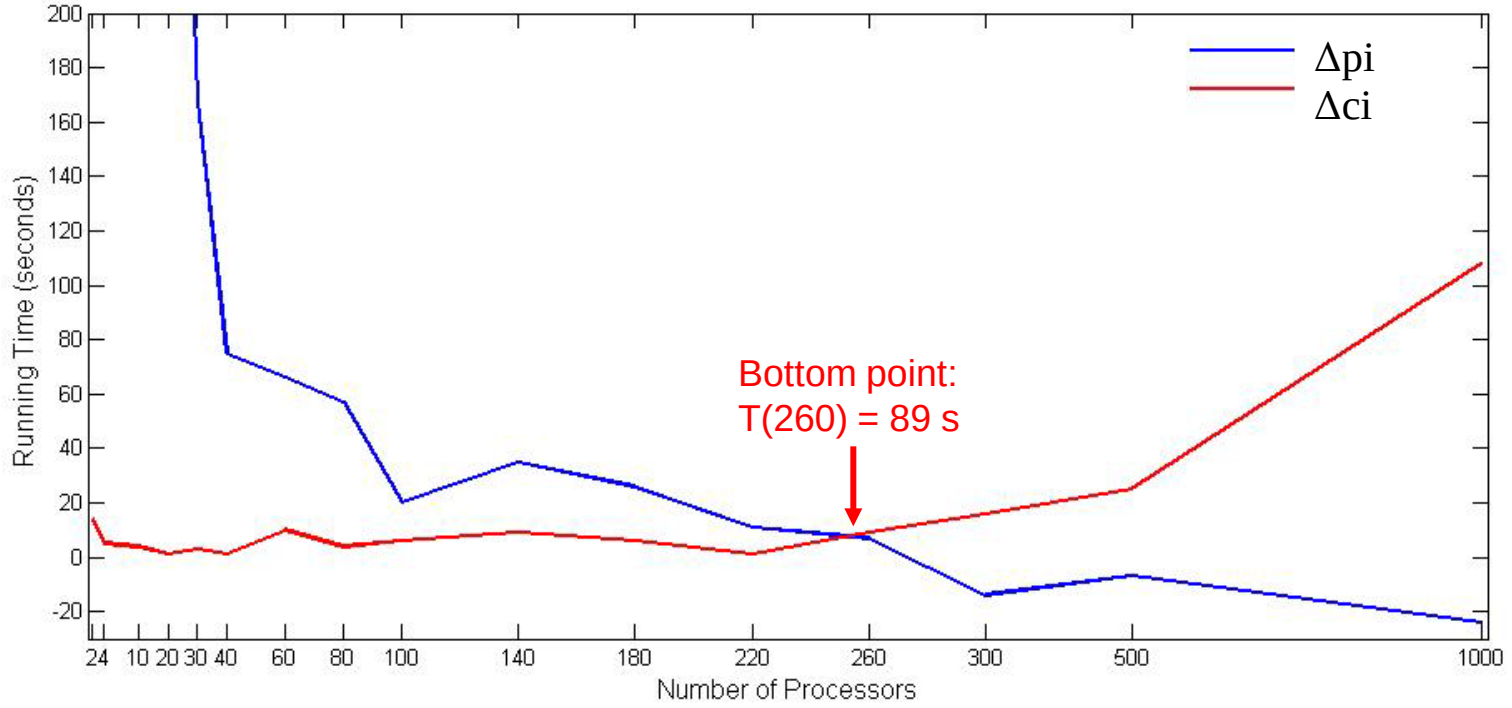
- $\Delta c_i = T(c_i) - T(c_{i-1})$
- $\Delta p_i = T(p_{i-1}) - T(p_i)$

Incremental running time:

$$T(i) - T(i-1) = (T(s) + T(c_i) + T(p_i)) - (T(s) + T(c_{i-1}) + T(p_{i-1}))$$
$$= \Delta c_i - \Delta p_i$$

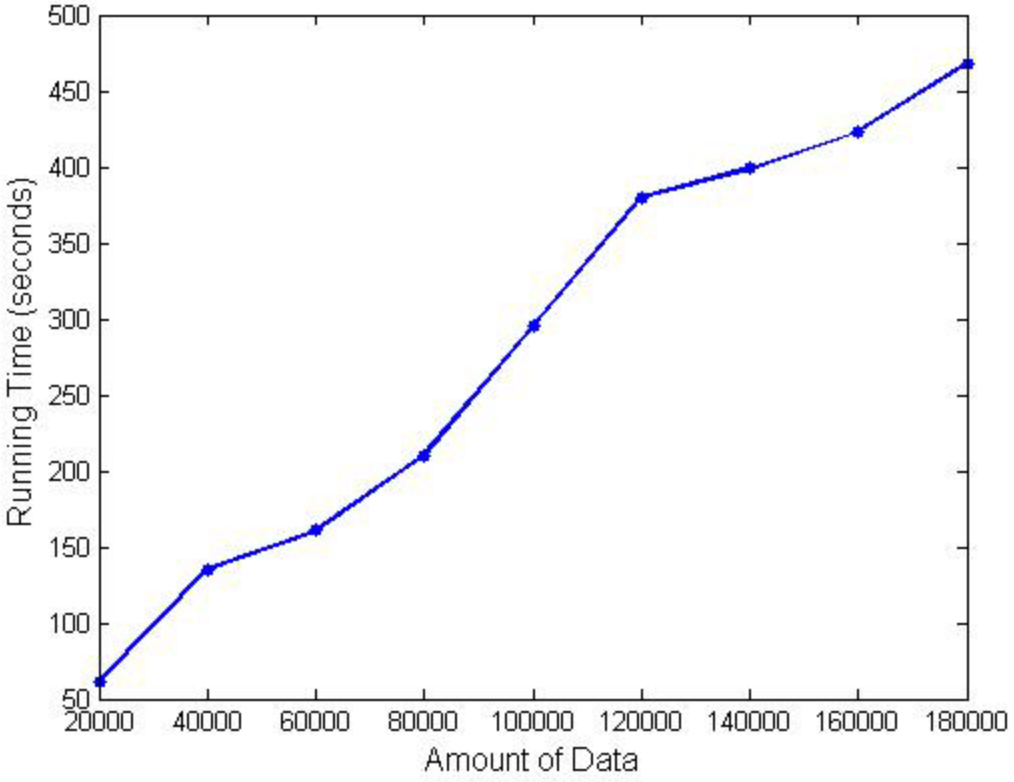
➡ If $\Delta c_i < \Delta p_i$, running time decreases
If $\Delta c_i > \Delta p_i$, running time increases

Δp_i and Δc_i



Running time with varying amount of data

Experiment Setup: 20 processors



Results

Input data set: 180,000 minutiae from NIST 4