Universitatea "Ştefan cel Mare" Suceava
Facultatea de Inginerie Electrică şi Ştiinţa Calculatoarelor

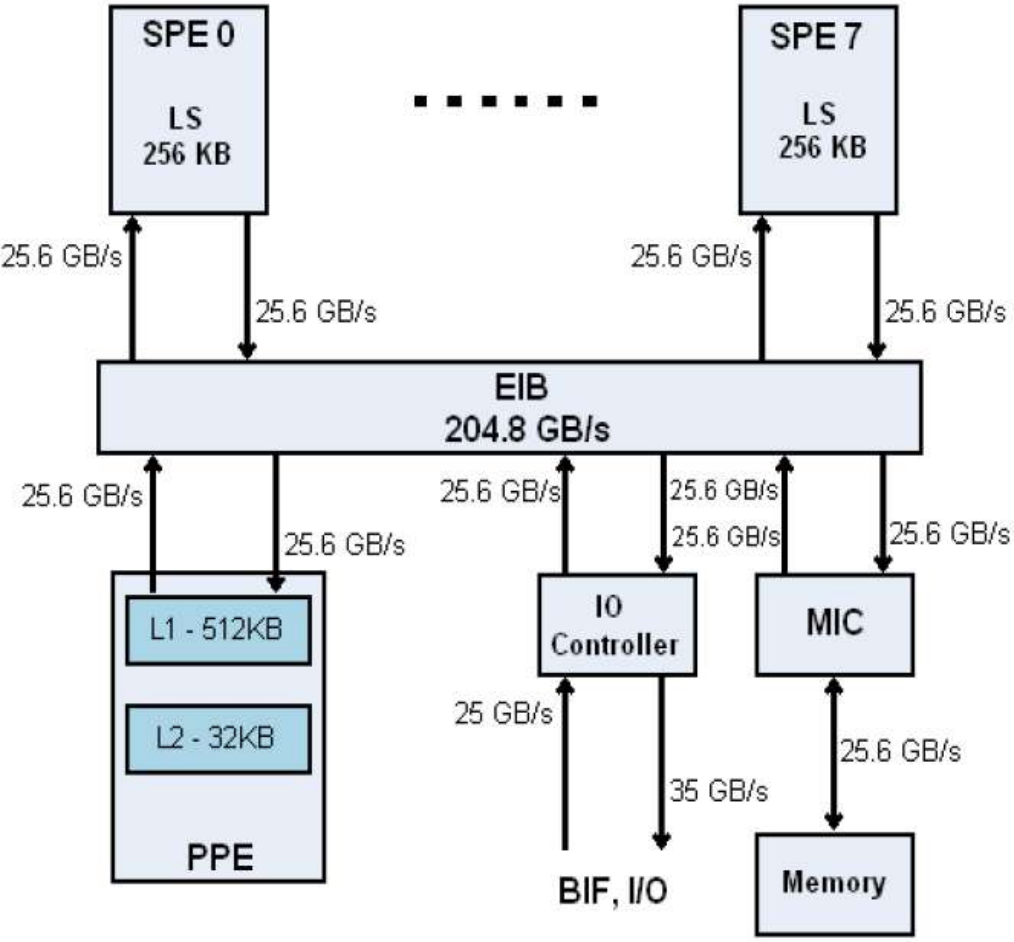
HIGH-PERFORMANCE COMPUTING ON A SUPERCOMPUTER BASED ON NEW-GENERATION PROCESSORS

Ioan Ungurean, Ionela Rusu, Stefan-Gheorghe Pentiuc
ioanu@eed.usv.ro, ionelar@eed.usv.ro, pentiuc@eed.usv.ro

OUTLINE

1. INTRODUCTION
2. USV-ROADRUNNER SUPERCOMPUTER
3. DETERMINATION OF A MODEL FOR A TYPE-3 CLASSIFIER
4. PARALLELIZATION OF THE ALGORITHM FOR USV-ROADRUNNER SUPERCOMPUTER
5. THE ACTIVATION OF THE SPE CORES
6. EXPERIMENTAL RESULTS
7. CONCLUSIONS

1. INTRODUCTION



- Cell Broadband Engine Architecture was developed by IBM for Cell B.E. processor from PlayStation 3 game console.
- This architecture was optimized for math intensive computing.

2. USV-ROADRUNNER SUPERCOMPUTER(1 /3)

- In December 2009, the HPC lab was completed with an IBM Roadrunner-type cluster, with a peak performance of 9.98 TFlops.
- Based on Cell BE processors
- Energy efficient (~337MFlops/watt)
- Architecture optimized for intensive arithmetic computations

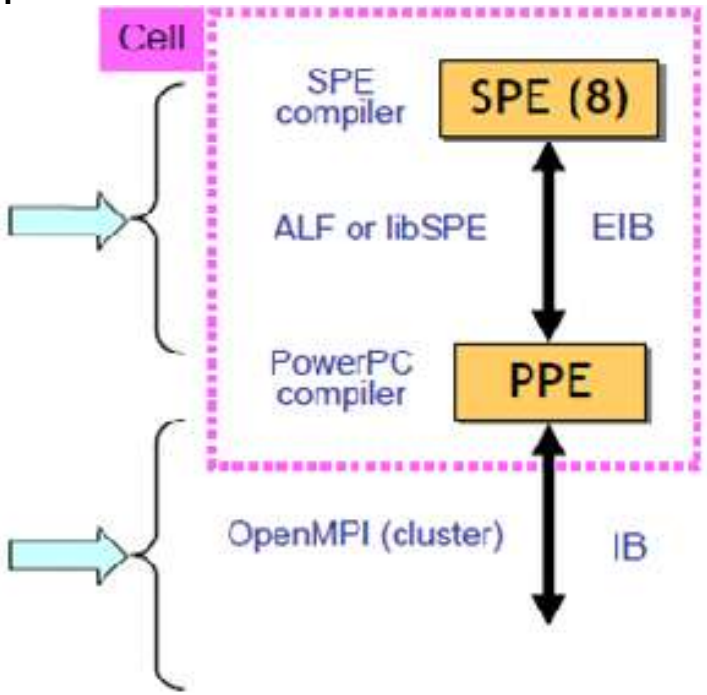
2. USV-ROADRUNNER SUPERCOMPUTER(2/3)

- 48 processing QS22 dual-CPU blades with Cell BE processors:
  - 2 IBM PowerXCell 8i CPUs (3.2 GHz)
  - 8 GB RAM per blade
  - 8 GB modular SSD
  - Gigabit Ethernet interface for administration
  - Infiniband 4xDDR interface for data transfer
  - GPFS access on the storage system for concurrent parallel access of files

2. USV-ROADRUNNER SUPERCOMPUTER(3/3)

- The applications development for USV-RoadRunner system must take into account that there are two levels of parallelization.
  - On the first level, the distribution of computation tasks to PowerXCell 8i processors (the core PPE) is performed.
  - On the second level, the distribution of computation tasks to SPE cores is performed.

1) Parallel computing (Cell BE)
   - *data partitioning;*
   - *sending to pipeline de tasks queue;*
   - *threads management and synchronization;*
2) Remote communication to/from Cell
   - *communications;*
   - *data synchronization;*
   - *process management and synchronization;*
   - *offloading intensive computations;*
3) MPI is the foundation



3. DETERMINATION OF A MODEL FOR A TYPE-3 CLASSIFIER (1 /2)

- The CLASS algorithm is part of the supervised classification algorithms.

- This means that the algorithm needs a training set, which is used for classification of patterns from the input set.

3. DETERMINATION OF A MODEL FOR A TYPE-3 CLASSIFIER (2/2)

Algorithm 1. Sequential algorithm.

**) Read the input data from a file.*

**) Allocate memory for the A(m x (p+1)) matrix.*

**) Allocate memory for the miu(m) vector.*

**) Initialize with 0 the elements of matrix A.*

for** i=1,n **do

for** j=1,p **do

A[CLASA[i]][j] += ANTREN[i][j]

end for

miu[CLASA[i]] += F[i]

end for

for** l=1,M **do

for** j=1,p **do

A[l][j] /= miu[l];

*A[l][p+1] +=A[l][j]*A[l][j]*(-1/2)*

end for

end for

end.

4. PARALLELIZATION OF THE ALGORITHM FOR USV-ROADRUNNER SUPERCOMPUTER

Algorithm 2. Parallelization of the algorithm for USV-RoadRunner supercomputer.

Read the input data from a file.

**) Allocate memory for the A(m x (p+1)) matrix.*

**) Allocate memory for the miu(m) vector.*

**) Initialize with 0 the elements of matrix A.*

**) Distribute the set of training patterns to the MPI process (nLolcal – the number of patterns assigned to the current MPI process).*

```
for i=1,nLocal do
    for j=1,p do
        A[CLASA[i]][j] += ANTREN[i][j]
    end for
    miu[CLASA[i]] += F[i]
end for
```

***)** Use MPI_AllReduce() function in order to update the matrix A and vector miu on all MPI processes from the system.*

```
for l=1,M do
    for j=1,p do
        A[l][j] /= miu[l];
        A[l][p+1] +=A[l][j]*A[l][j]*(-1/2)
    end for
end for
```

***)** Use MPI_AllReduce() function in order to update the matrix A on all MPI processes from the system.*

end.

5. THE ACTIVATION OF THE SPE CORES

Algorithm 3. The activation of the SPE cores.

**) Read the input data from a file.*

**) Allocate memory for the A(m x (p+1)) matrix.*

**) Allocate memory for the miu(m) vector.*

**) Initialize with 0 the elements of matrix A.*

**) Distribute the set of training patterns to the MPI process (nLocal– the number of patterns assigned to the current MPI process).*

Distribute the local set of training patterns to the SPE cores

**) Send a signal to the SPE cores in order to begin the compute of matrix A based on assigned patterns.*

**) Wait for a mailbox from each SPE core, which signals the end of assigned jobs.*

for** l=1,M **do

for** j=1,p **do

A[l][j] /= miu[l];

*A[l][p+1] +=A[l][j]*A[l][j]*(-1/2)*

end for

end for

****)** Use MPI_AllReduce() function in order to update the matrix A and vector miu on all MPI processes from the system.*

end.

6. EXPERIMENTAL RESULTS (1 /4)

- In order to test the performance of this algorithm, we have chosen the p53 mutants dataset.
- This dataset represents the models of the mutant p53 protein, models that can be used to predict the transcriptional activity of p53 protein (cancer diagnosis).
- This data set has 16,722 patterns, each pattern with 5409 features.

6. EXPERIMENTAL RESULTS (2/4)

- In first step, the performances were measured by execution of the algorithm on one processor PowerXCell 8i, without use of the SPE cores.
- Next, the number of PowerXCell 8i processor was doubled until the maximum number (96) (see Fig. 2).

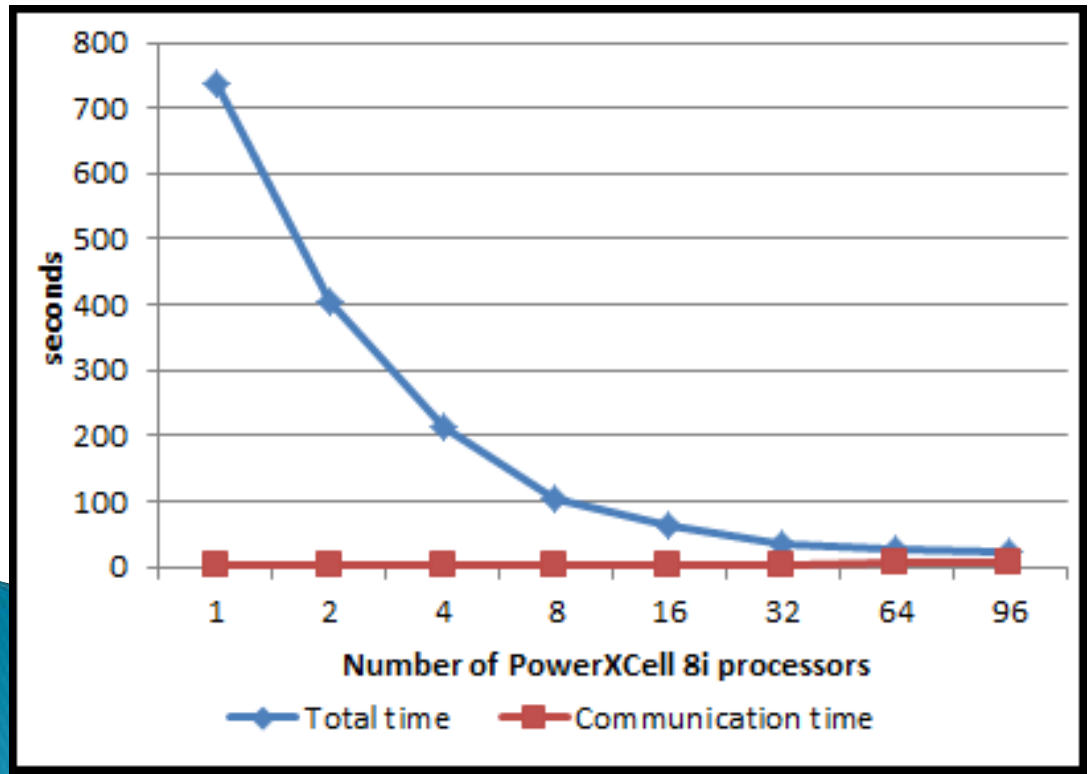


Figure 2. The performances obtained without use of SPE cores.

6. EXPERIMENTAL RESULTS (3/4)

- Fig 3 presents the total execution time and communication time for the case in which all 8 SPE cores on each PowerXCell 8i processor are used.

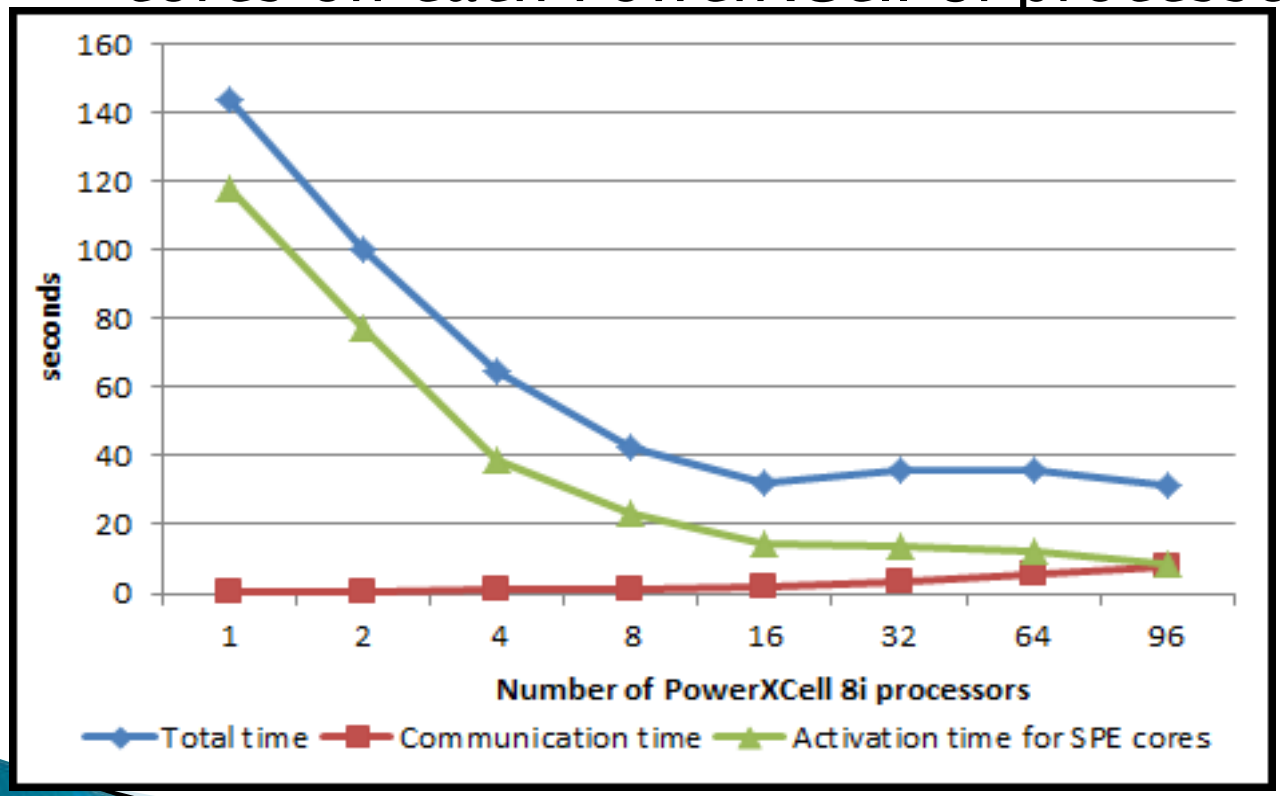


Figure 3. The performances obtained with use of SPE cores.

6. EXPERIMENTAL RESULTS (4/4)

- Fig 4 presents the speeds-up obtained by increasing the number of PowerXCell 8i processors with and without use of the SPE cores.
- Speeds-ups are reported to the algorithm execution on a PowerXCell 8i processor without use of the SPE cores.

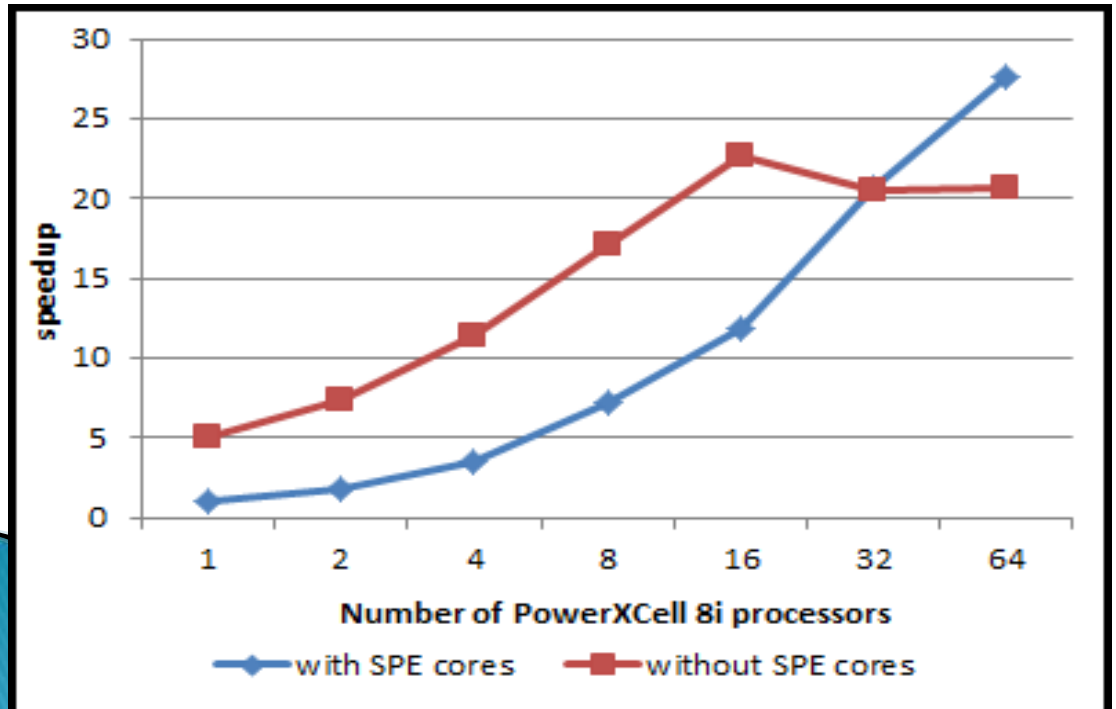


Figure 4. Graphical representation of speeds-up obtained.

7. CONCLUSIONS

- In this paper, we present the methods used in order to design, program, and analyze a multilevel parallel variant so as to determine a model for a type-3 classifier.

- A comparison was performed between the sequential variant and parallelized variant was carried out on a single level, as well as for the parallelized variant on two levels.

7. CONCLUSIONS

- The performances achieved by algorithm execution were analyzed for the USV-RoadRunner supercomputer. For the moment, we obtain a speed-up of 28.

- Algorithm can be further optimized by using the SIMD facilities held of SPE cores. Using these facilities, the speedup can increase with an order size (x10).

Universitatea "Ştefan cel Mare" Suceava

Facultatea de Inginerie Electrică şi Ştiinţa Calculatoarelor

Thank you!