A Simulated Annealing algorithm for GPU clusters

Maciej Zbierski

Institute of Computer Science
Warsaw University of Technology

Parallel Processing and Applied Mathematics 2011

Simulated Annealing (1)

- Metaheuristic for the global optimization problem
- Adaptation of Metropolis-Hastings
- Effective for computationally hard (NP-hard) problems

Simulated Annealing (2)

Metropolis-Hastings algorithm

- On start: pick $\mathbf{x}_{min} = \mathbf{x}_0 \in \mathbb{R}^n$, $\mathbf{L} < \mathbf{x}_0 < \mathbf{U}$
- Find $\mathbf{x}_{i+1}$ based on $\mathbf{x}_{min}$
- If $f(\mathbf{x}_{i+1}) < f(\mathbf{x}_{min})$ then $\mathbf{x}_{min} = \mathbf{x}_{i+1}$
- Else $\mathbf{x}_{min} = \mathbf{x}_{i+1}$ with probability p

Simulated Annealing (2)

Metropolis-Hastings algorithm

- On start: pick $\mathbf{x}_{min} = \mathbf{x}_0 \in \mathbb{R}^n$, $\mathbf{L} < \mathbf{x}_0 < \mathbf{U}$
- Find $\mathbf{x}_{i+1}$ based on $\mathbf{x}_{min}$
- If $f(\mathbf{x}_{i+1}) < f(\mathbf{x}_{min})$ then $\mathbf{x}_{min} = \mathbf{x}_{i+1}$
- Else $\mathbf{x}_{min} = \mathbf{x}_{i+1}$ with probability p

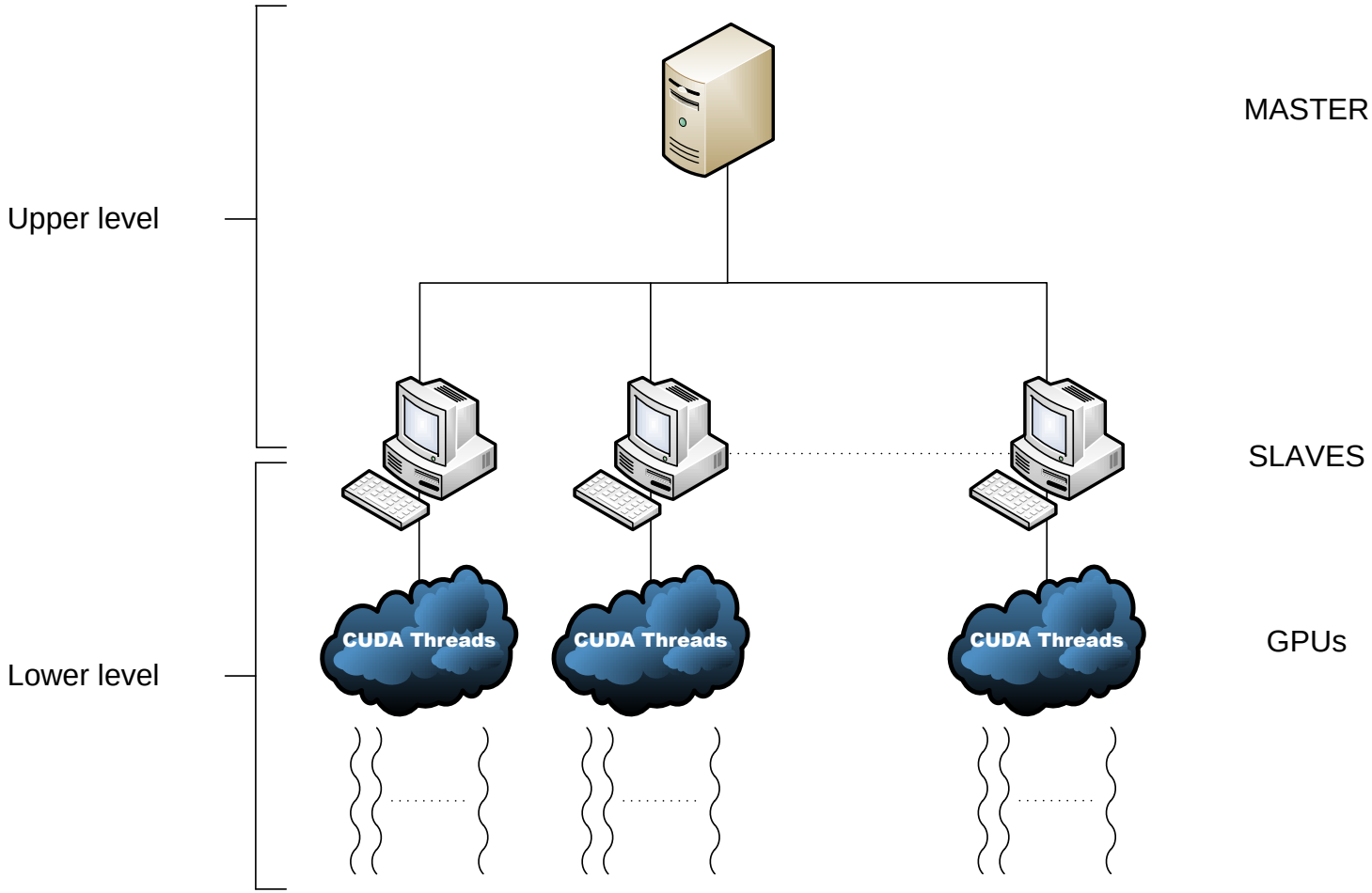
Simulated Annealing

- The p value depends on the temperature
- The temperature is reduced whenever the suboptimal solution is accepted

Basic concepts of the processing platform (1)

- Two-level hierarchical architecture (master-slave)

Basic concepts of the processing platform (1)



Basic concepts of the processing platform (2)

Assumption:

- Asynchronous communication on the upper level

Basic concepts of the processing platform (2)

Assumption:

- Asynchronous communication on the upper level

Implications:

- No direct cooperation between nodes (nor GPU threads)

Basic concepts of the processing platform (2)

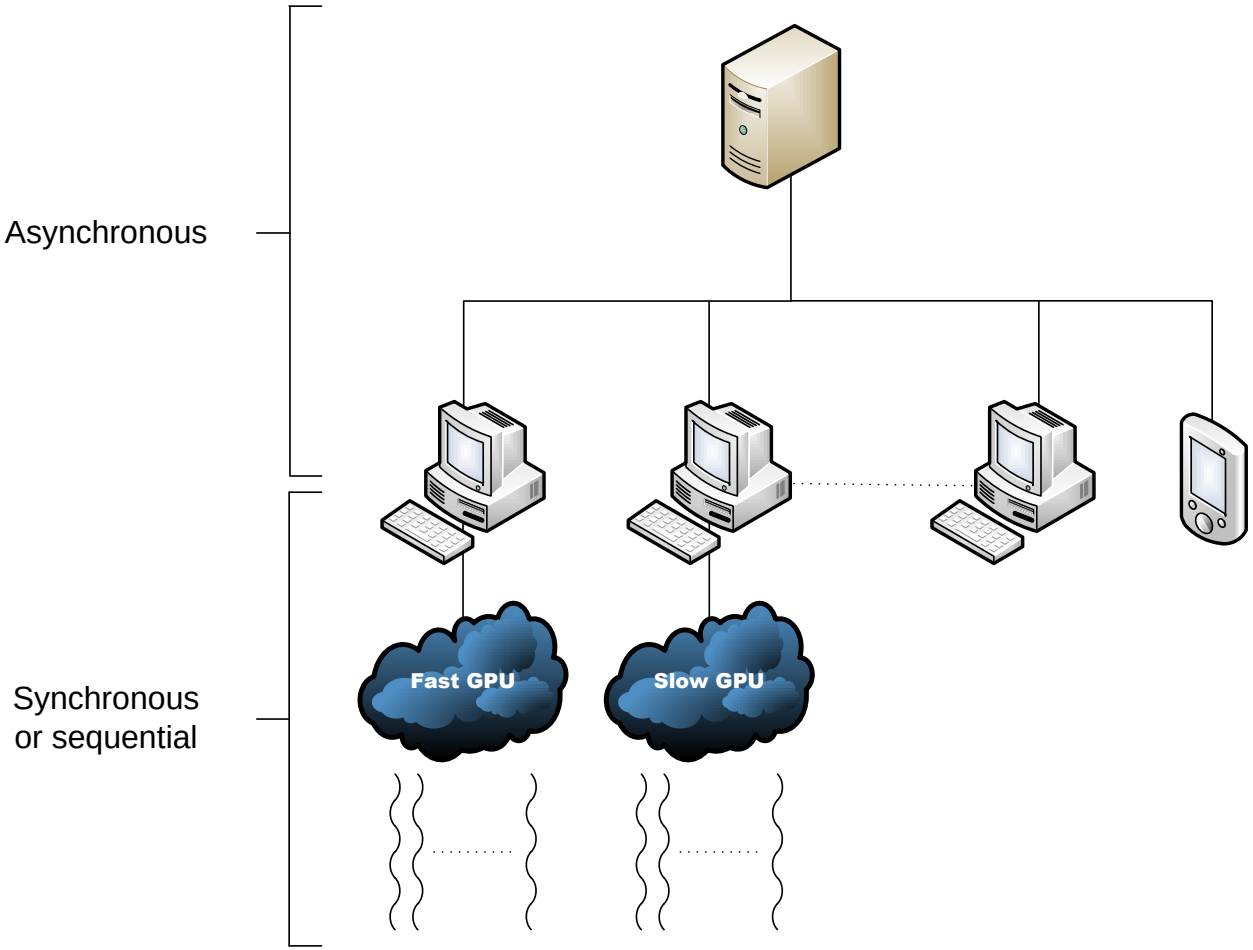
Assumption:

- Asynchronous communication on the upper level

Implications:

- No direct cooperation between nodes (nor GPU threads)
- But: the system may be heterogeneous

Basic concepts of the processing platform (2)



The lower level

- Initial solution computed on CPU
- Each thread perform m steps of sequential SA modifying only the d-th dimension ($d = ID \bmod n$)
- All solutions are adaptively composed into the new one

The upper level

Whenever a result is obtained:

- The result is adaptively combined to the previous one
- If the new result is better, it is accepted

Test environment

- Comparison with traditional parallel SA (MHCS)
- MHCS: Sun Fire X4600M2 (8x AMD Opteron 8218)
- TLSA: 16x NVIDIA GeForce 130 GT
- n-dimensional test functions ($n \in \{10, 100\}$)

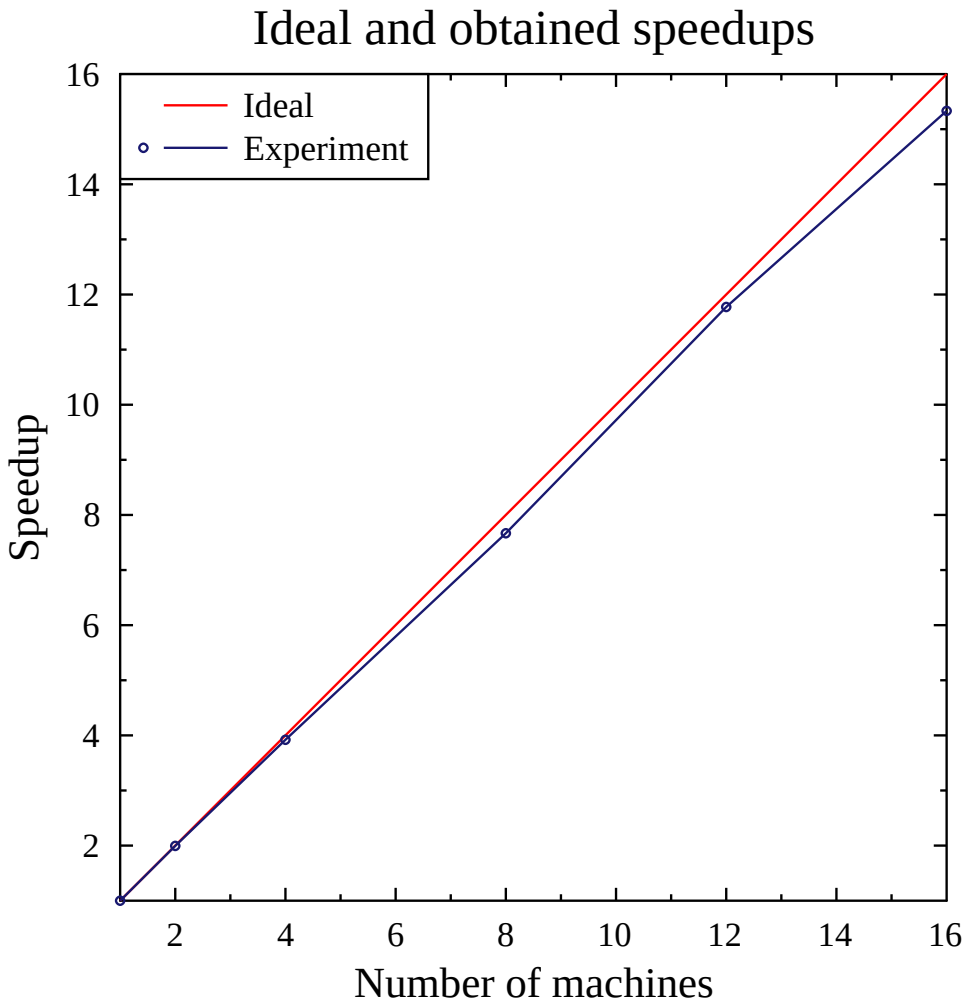
Time results, $n = 10$ (mean of 5 runs)

Function	**GPU cluster** [s]	**Sun Fire** [s]
Ackley	26.36	> 4min
De Jong	14.65	1.11
Giewangk	26.38	> 4min
Michalewicz	26.29	32.94
Rastrigin	26.28	> 4min
Rosenbrock	34.92	1.32
Schwefel	26.42	46.47

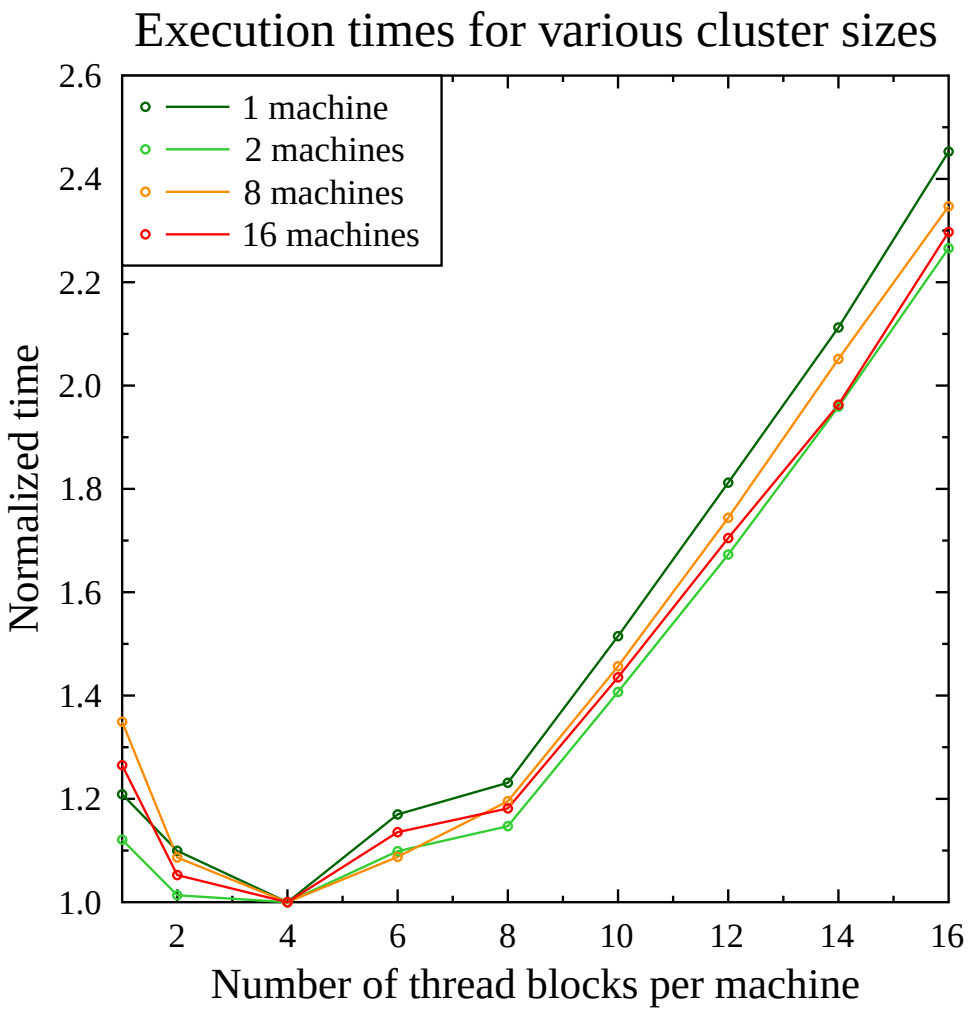
Time results, $n = 100$ (mean of 5 runs)

Function	**GPU cluster** [s]	**Sun Fire** [s]
Ackley	109.82	704.56
De Jong	55.15	113.98
Giewangk	110.64	7886.96
Michalewicz	113.41	$>$ 4h
Rastrigin	109.22	174.93
Rosenbrock	165.14	1284.03
Schwefel	111.81	1027.48

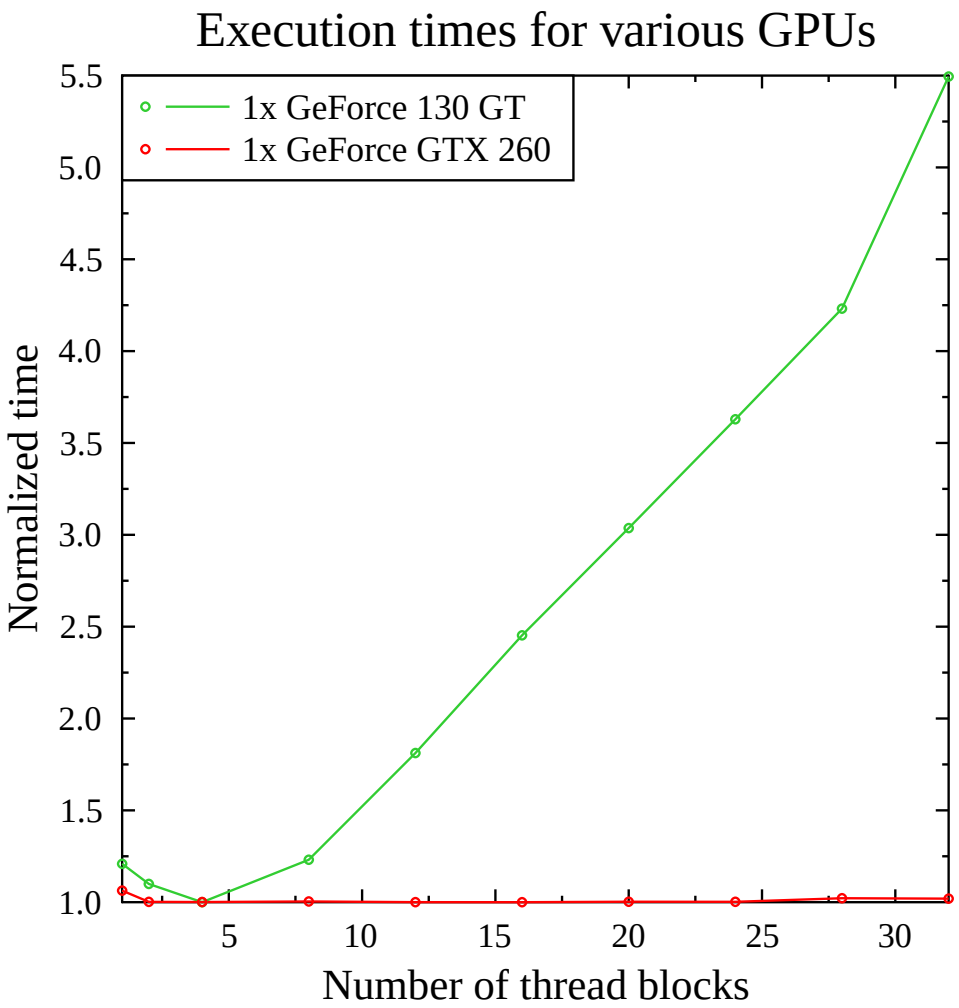
Scalability (1)



Scalability (2)



Scalability (3)



Conclusion

- Good efficiency for complex problems
- Higher versatility (heterogeneous environment)
- Almost ideal scalability w.r.t. number of machines (up to some point)

Thank you!

Time results, $n = 10$

Function	GPU cluster		Sun Fire	
	m [s]	σ [s]	m [s]	σ [s]
Ackley	26.36	0.31	$>$ 4min	0
De Jong	14.65	0.46	1.11	0.55
Giewangk	26.38	0.29	$>$ 4min	0
Michalewicz	26.29	0.35	32.94	10.39
Rastrigin	26.28	0.22	$>$ 4min	0
Rosenbrock	34.92	0.29	1.32	0.92
Schwefel	26.42	0.25	46.47	18.01

Time results, $n = 100$

Function	GPU cluster		Sun Fire	
	m [s]	σ [s]	m [s]	σ [s]
Ackley	109.82	0.1	704.56	302.57
De Jong	55.15	0.33	113.98	5.61
Giewangk	110.64	0.33	7886.96	7522.44
Michalewicz	113.41	0.27	$>$ 4h	0
Rastrigin	109.22	0.12	174.93	19.17
Rosenbrock	165.14	0.27	1284.03	1479.14
Schwefel	111.81	0.26	1027.48	70.75

Results, $n = 100$

Function		GPU Cluster		Sun Fire	
name	**minimum**	**m**	σ	**m**	σ
Ackley	0	0.0057044	0.0009475	2.5719460	0.1417264
De Jong	0	0.0002648	0.0001606	0.0257231	0.0101550
Giewangk	0	0.0091233	0.0081603	3.6563780	0.7952309
Michalewicz	?	−99.60930	0.0029908	−93.37772	0.7084963
Rastrigin	0	0.0025821	0.0015947	0.0047532	0.0005984
Rosenbrock	0	107.64608	26.121126	91.917120	56.921374
Schwefel	−41898.29	−41897.96	0.0547723	−38844.24	290.62179

Results, $n = 10$

Function		GPU Cluster		Sun Fire	
name	**minimum**	**m**	σ	**m**	σ
Ackley	0	0.0004817	0.0002419	0.0030275	0.0007530
De Jong	0	0.0000120	0.0000107	0.0000086	0.0000037
Giewangk	0	0.0000078	0.0000133	0.0536091	0.0249136
Michalewicz	?	−9.660150	0	−9.660132	0.0000130
Rastrigin	0	0.0000178	0.0000298	0.0006522	0.0002632
Rosenbrock	0	0.3762460	0.4398527	0.1054340	0.1191452
Schwefel	−4189.829	−4183.096	3.4085745	−4141.278	23.645863

Observations

- More threads → more SA steps in parallel
- Best when the number of dimensions ≈ number of threads
- What with other cases?