Large scale performance prediction

Raphaël Couturier

FEMTO-ST, University of Franche-Comte, France

14/09/2015

Outline

1. Motivations
2. Context
3. Our approach
4. Experiments
5. Conclusion & perspectives

AND team

- Numerical distributed algorithm
- High performance computing: numerical methods, asynchronous algorithms
- Wireless sensor networks: data reduction with periodic sensors
- Security: steganography, watermarking
- Bio-informatic: ancestral virus mutation

Motivations for predictions of scalability 1/2

- Scalability of applications:
  - understand the limitation: improve the code
  - chose the appropriate architecture
  - access to supercomputer is expensive
- Strong scalability
- Weak scalability
- Large scale scalability is the graal

Motivations for predictions of scalability 2/2

- Networks parameters are complex
- Making models is very difficult:
  - abstraction of the code
  - abstraction of the system
- Best approach: Paraver Dimemas (BSC Barcelona), trace analysis

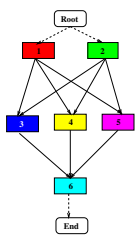
Context

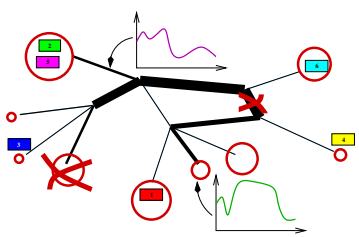
- Weak scaling
- Use real executions with few cores (for example 256 and 512) for measuring the communication and the computation
- Build a model that simulates the same application
- Extrapolate the performance with the model
- Evaluate the prediction

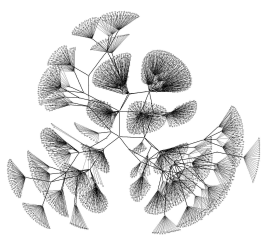
SimGrid 1/2

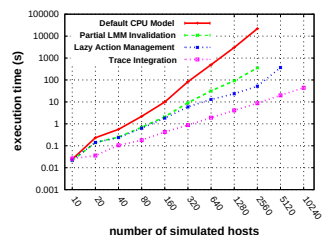
- Tool for reproducible simulations
- Simulation of distributed systems composed of heterogenous machines and networks
- Comfortable for users
  - Get preliminary results from partial implementations
  - Experimental campaign with thousands of runs within the week
  - Test your scientific idea, don't fiddle with technical subtleties (yet)

Idea to test + Experimental setup + Simulation model ⇒ Scientific results









SimGrid 2/2

- SMPI: compilation of MPI program
- Simulation within SimGrid
- With different platforms, we obtain different results
- Difficulty to choose a platform close to reality:
  - processor architecture (core, cache, bus, ...)
  - network, switch
  - MPI implementation
  - behavior of the operating system

Our approach

- Run a real code on the machine (for example with 256 and 512 cores)
- Measure the computation and the communication times
- Build a model in SimGrid such that an excution of a smaller problem size with the same code give similar results
- Increase the size in SimGrid
- Make prediction of the scalability by extrapolation

Some elements

- With the weak scaling case, only the communications matter
- If the model in SimGrid is good, it should predict the scalability of the communications
- Necessity to provide automatic procedures

GMRES and Krylov multisplitting

- 2 codes: standard GMRES, and a multisplitting code to solve large sparse linear systems
- Krylov multisplitting: 2 iterations method
  - bloc Jacobi method like (with GMRES inside each bloc)
  - Krylov residual minimization each 10 outer iterations (for ex.)
- Number of iterations dependent of the problem size (computed with a small cluster)

Experiments

- Architecture: Curie in CEA 80640 cores (16 cores per processors)
- Network: Infiniband QDR
- Influence of other users is more significant when the number of processors increases

Experiments 1/3

x256	# of block	# of cores	# of iterations	Communication time per iteration (ms)		Computing time per iteration (ms)	Total execution time (s)		Prediction error (%)
				Measured	Predicted		Measured	Predicted	
GMRES	2	512	11,204	0.49	0.50	2.29	31.21	31.41	**0.66**
	4	1,024	17,858	0.60	0.61	2.32	52.13	52.20	**0.14**
	8	2,048	28,391	0.77	0.67	2.34	88.37	84.52	**4.35**
	16	4,096	45,405	0.82	0.79	2.27	140.34	140.63	**0.21**
	32	8,192	72,560	0.85	0.84	2.28	227.12	228.54	**0.62**
	64	16,384	116,318	1.06	0.96	2.35	396.19	380.19	**4.04**
					avg	2.31			
MULTI	2	512	4,160	0.71	0.68	2.78	14.49	14.5	**0.02**
	4	1,024	5,320	0.74	0.74	2.82	18.98	18.9	**0.60**
	8	2,048	10,760	0.87	0.82	2.82	39.75	38.9	**2.10**
	16	4,096	20,640	0.78	0.83	2.77	73.29	75.0	**2.33**
	32	8,192	42,840	0.83	0.86	2.79	154.99	156.9	**1.21**
	64	16,384	65,400	1.12	0.93	2.83	258.47	243.7	**5.70**
					avg	2.80			

Experiments 2/3

x512	# of block	# of cores	# of iterations	Communication time per iteration (ms)		Computing time per iteration (ms)	Total execution time (s)		Prediction error (%)
				Measured	Predicted		Measured	Predicted	
GMRES	2	1,024	17,858	0.61	0.61	2.32	52.3	52.2	**0.20**
	4	2,048	28,391	0.75	0.77	2.34	87.7	87.5	**0.29**
	8	4,096	45,405	0.84	0.82	2.27	141.2	142.5	**0.88**
	16	8,192	72,560	0.85	0.99	2.29	227.5	239.3	**5.22**
	32	16,384	116,318	1.03	1.04	2.35	393.2	390.1	**0.78**
					avg	2.31			
MULTI	2	1,024	7,080	0.88	0.89	2.81	26.1	26.2	**0.28**
	4	2,048	12,400	0.99	0.97	2.84	47.6	46.8	**1.51**
	8	4,096	19,600	1.00	0.94	2.77	73.8	73.5	**0.49**
	16	8,192	33,960	0.86	1.10	2.79	123.8	132.8	**7.27**
	32	16,384	64,800	1.12	1.11	2.83	256.5	254.1	**0.92**
					avg	2.81			

Experiments 3/3

x1024	# of block	# of cores	# of iterations	Communication time per iteration (ms)		Computing time per iteration (ms)	Total execution time (s)		Prediction error (%)
				Measured	Predicted		Measured	Predicted	
GMRES	2	2,048	28,391	0.73	0.71	2.341908	87.2	85.9	**1.49**
	4	4,096	45,405	0.81	0.90	2.272290	140.1	145.8	**4.01**
	8	8,192	72,560	0.81	0.96	2.285581	224.8	237.7	**5.75**
	16	16,384	116,318	1.10	1.15	2.348193	400.7	402.5	**0.47**
					avg	*2.311993*			
MULTI	2	2,048	9,040	1.11	1.03	2.846920	35.7	34.6	**3.03**
	4	4,096	18,480	1.11	1.12	2.751901	75.2	72.6	**3.48**
	8	8,192	29,640	0.97	1.09	2.787762	111.5	115.4	**3.50**
	16	16,384	46,960	1.26	1.25	2.832391	192.3	190.4	**0.99**
					avg	*2.804744*			

Open questions

- What about the strong scaling case?
  $\Rightarrow$ The model should also simulate the computation
- What about irregular problems?
  $\Rightarrow$ No idea yet
- Generalization of the approach?
  $\Rightarrow$ We plan to make more experiments

Conclusion

- New method to make prediction of scalability
- The code is executed in the SimGrid model
- Good results with two applications

Perspectives

- Study the strong scaling approach
- Test with different applications:
  - real applications
  - bigger applications
  - irregular applications