Prediction of the scalability of weak scaling applications
Raphaël Couturier

For many applications, it may be difficult to predict how they will scale with 2,000 cores if we have the scalability for 100 cores, for example. We are interested in predicting the scalability of two iterative methods, with weak scaling, for solving large sparse linear systems. The two methods we want to compare are the standard GMRES and a Krylov multisplitting iterative solver also based on GMRES.
Our goal is to measure all the computation time and simulate the communication through SimGrid in order to predict the scalability of these two methods when the communication costs have been modelized into SimGrid. SimGrid is a tool that has successfully been used to simulate the behavior of an application before running it in real grid architectures.

In the following we give an example of what we can compute with the CURIE cluster, in France. We have measured the execution times and computations times of 256 cores with 2 and 4 blocks (so it corresponds to 512 and 1024 cores) for both our codes. Then we model that with Simgrid in order to have approximately the same results. Then we use Simgrid to predict the communication times with 8, 16, 32 and 64 blocks (so it corresponds to 2.048, 4.096, 8.192 and 16.384 cores). As the computing time per iteration is constant (due to the weak scaling use), we can predict the communication time and with the number of iterations computed with other execution with less processors, we can predict the total execution times. Finally we compare the estimated execution times to the real measured execution times to compute the prediction error. We can observe that the prediction error is low.

x256	# of block	# of cores	# of iterations	Communication time per iteration (ms)		Computing time per iteration (ms)	Total execution time (ms)		Prediction error (%)
				Measured	Predicted		Measured	Predicted	
GMRES	2	512	11204	0.4923092	0.4951461	2.293184	31208.7	31414.1	**0.66**
	4	1,024	17,858	0.6031661	0.6145477	2.315872	52128.2	52203.1	**0.14**
	8	2,048	28,391	0.7733767	0.6683101	2.339122	88367.0	84519.8	**4.35**
	16	4,096	45,405	0.8180651	0.7885642	2.272745	140338.2	140630.5	**0.21**
	32	8,192	72,560	0.8476061	0.8409706	2.282531	227122.7	228538.9	**0.62**
	64	16,384	116,318	1.0574628	0.9598873	2.348647	396191.9	380193.6161	**4.04**
					avg	*2.308683*			
MULTI	2	512	4160	0.7060143	0.682133	2.776880	14488.8	14491.2	**0.02**
	4	1024	5320	0.7439338	0.7440872	2.822878	18975.4	18861.6	**0.60**
	8	2048	10760	0.8725864	0.815259	2.821571	39749.1	38914.5	**2.10**
	16	4096	20640	0.7820521	0.832425	2.769015	73294.0	75000.7	**2.33**
	32	8,192	42,840	0.829202	0.8600562	2.788599	154,986.6	156,853.7	**1.20**
	64	16,384	65400	1.1230907	0.9255363	2.829032	258468.8	243737.0	**5.70**
					avg	*2.801329*			

In future works, we would like to investigate the prediction of strong scaling code. So compared to what we have done, we need to be able to predict the computing time and the communication time.