Raxml repeats benchmarks

April 13, 2017

1 Experiments context

We computed the following results using Haswell cluster, with raxml-ng (branch dev, commit 2615802f67d51c9d3ea2d08c570d42b30d4f2a28, + some code modification to use repeats).

We run raxml-ng with the following arguments:

```
$ LD_PRELOAD=libjemalloc.so mpirun ./raxml-ng-mpi --seed=43
 --msa data.phy --model data.part
 --simd AVX --threads 1  --search --repeats on
$ LD_PRELOAD=libjemalloc.so mpirun ./raxml-ng-mpi --seed=43
 --msa data.phy --model data.part
 --simd AVX --threads 1  --search
```

We used the following datasets (sites is the number of unique sites after pattern compression):

dataset	taxas	sites	partitions	type
404	404	7444	11	DNA
1kite_science	144	371434	50	DNA
1kite_hyme	174	2248590	4116	DNA
Antl_1_1_nt	40	522173	658	DNA
para_1_nt	193	1514275	3714	DNA
sativa_in	973	2477	1	DNA

2 Execution time, scalability and repeats speedups

- time indicates the elapsed time of the whole run
- time*threads is the same quantity mutliplied by the number of threads
- speedup is the ratio between time repeats and time

1kyte_hyme

threads	sites/thread	time	time repeats	time*threads	time*threads repeats	speedup
64	35134	31295s	22780s	2002880s	1457920s	1.37
128	17567	16570s	11880s	2120960s	1520640s	1.39
256	8783	9067s	6669s	2321152s	1707264s	1.35
512	4391	5006s	4002s	2563072s	2049024s	1.25

404

threads	sites/thread	time	time repeats	time*threads	time*threads repeats	speedup
1	7444	5760s	3910s	5760s	3910s	1.47
2	3722	2718s	2049s	5436s	4098s	1.32
4	1861	1883s	1208s	7532s	4832s	1.55
8	930	1234s	693s	9872s	5544s	1.78
16	465	732s	443s	11712s	7088s	1.65

antl_1_1_nt2

threads	sites/thread	time	time repeats	time*threads	time*threads repeats	speedup
16	32635	1651s	1255s	26416s	20080s	1.31
64	8158	394s	297s	25216s	19008s	1.32
256	2039	118s	5524s	30208s	1414144s	.02
512	1019	92s	188s	47104s	96256s	.48

kyte

threads	sites/thread	time	time repeats	time*threads	time*threads repeats	speedup
64	5803	2246s	1656s	143744s	105984s	1.35
128	2901	1055s	828s	135040s	105984s	1.27
256	1450	717s	520s	183552s	133120s	1.37

para_1_nt

threads	sites/thread	time	time repeats	time*threads	time*threads repeats	speedup
128	11830	12601s	9096s	1612928s	1164288s	1.38
256	5915	6114s	4529s	1565184s	1159424s	1.34
512	2957	3599s	2830s	1842688s	1448960s	1.27
1024	1478	2796s	2341s	2863104s	2397184s	1.19

sativa

threads	sites/thread	time	time repeats	time*threads	time*threads repeats	speedup
2	1238	6919s	4455s	13838s	8910s	1.55
4	619	4624s	2856s	18496s	11424s	1.61

3 Load balancing and time spent in reduce

- bad LB indicates the percentage of the time we lose because of bad load balancing. It can be seen as the time we could save with an optimal load balancing.
- reduce indicates the percentage of the time we spend in the reduction operation
- the sum of bad LB and reduce is the total time we lose because of parallelisation

1kyte_hyme

threads	sites/thread	bad LB	bad LB repeats	reduce	reduce repeats
64	35134	0.78%	1.43%	1.62%	2.39%
128	17567	1.23%	1.47%	3.58%	5.06%
256	8783	4.37%	1.87%	8.13%	11.5%
512	4391	2.70%	2.00%	16.6%	21.2%

404

threads	sites/thread	bad LB	bad LB repeats	reduce	reduce repeats
1	7444	-nan%	-nan%	0%%	0%%
2	3722	0.27%	0.34%	.714%	1.46%
4	1861	2.13%	2.89%	2.36%	3.01%
8	930	4.31%	4.60%	3.69%	5.01%
16	465	6.49%	7.43%	6.71%	10.0%

antl_1_1_nt2

threads	sites/thread	bad LB	bad LB repeats	reduce	reduce repeats
16	32635	0.43%	0.90%	1.04%	1.38%
64	8158	2.04%	2.25%	4.30%	6.10%
256	2039	4.36%	0.06%	13.5%	98.4%
512	1019	5.04%	1.54%	27.1%	70.1%

kyte

threads	sites/thread	bad LB	bad LB repeats	reduce	reduce repeats
64	5803	2.02%	8.35%	6.87%	9.41%
128	2901	3.33%	10.7%	9.44%	11.4%
256	1450	4.19%	10.1%	30.7%	26.0%

para_1_nt

threads	sites/thread	bad LB	bad LB repeats	reduce	reduce repeats
128	11830	1.81%	1.86%	5.70%	6.78%
256	5915	2.21%	2.24%	11.4%	16.2%
512	2957	2.51%	2.50%	23.9%	29.4%
1024	1478	3.91%	2.28%	39.5%	45.9%

sativa

threads	sites/thread	bad LB	bad LB repeats	reduce	reduce repeats
2	1238	0.60%	4.97%	1.22%	2.24%
4	619	2.46%	6.08%	2.37%	4.56%