Alperen Karaoglu N13012011 ak4410

Lab 1 Report

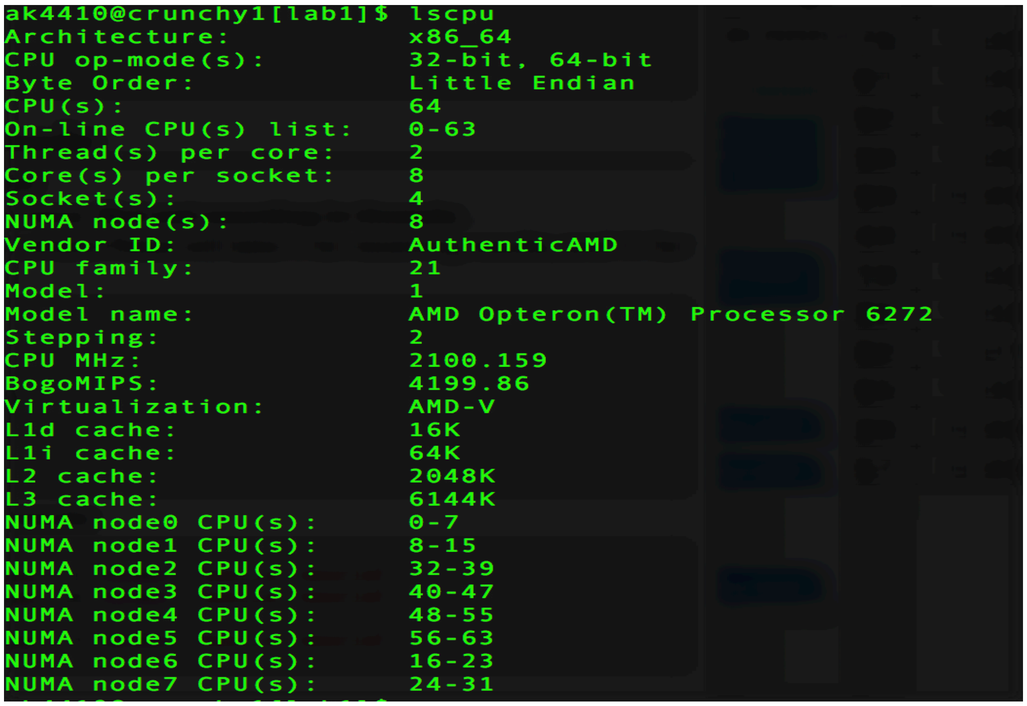
#4 Average output of the Time command

	1	2	4	8	16	32	64
2	0.382	0.3826					
4	0.3902	0.3834	0.3816				
8	0.39	0.38	0.382	0.3826			
16	0.383	0.384	0.3838	0.3864	0.3854		
32	0.3844	0.3848	0.3832	0.3946	0.3844	0.6852	
64	0.3824	0.3814	0.385	0.3886	0.3838	0.6806	0.9858
128	0.3776	0.3714	0.3718	0.3772	0.387	0.6786	0.984
256	0.382	0.3792	0.3848	0.3834	0.3818	0.6896	0.992
512	0.3862	0.3794	0.3882	0.3828	0.3836	0.681	0.9874
1024	0.3866	0.384	0.3818	0.385	0.3872	0.6882	0.9828
2048	0.3878	0.3858	0.382	0.3852	0.3842	0.6838	0.992200
4096	0.3834	0.386600	0.385	0.384	0.3826	0.6922	0.9886

#5 Speedup relative to the number of Processes

	1	2	4	8	16	32	64
2	0.999476	1.008887					
4	0.986161	1.001565	1.011006				
8	0.987692	1.005789	1.008377	0.992682			
16	0.994256	1.008333	0.999479	0.98913	0.998443		
32	0.997399	0.994802	1.000522	0.972124	1.003642	0.553999	
64	1.028243	0.995281	1.001558	1.018013	0.98593	0.562739	0.383851
128	1.002119	1.024233	1.003228	0.988335	0.971059	0.551429	0.378049
256	0.973822	1.021097	1.002599	1.013041	1.002619	0.555684	0.385685
512	0.999482	1.012652	0.992787	0.996865	1.012513	0.564464	0.390115
1024	0.998965	1.009375	1.007858	0.985455	0.992769	0.558849	0.394587
2048	1.004126	0.991706	1.008377	1.005192	1.003123	0.56771	0.386011
4096	0.998957	0.997413	0.993766	0.999479	1.011500	0.561687	0.388833

#6



```
ak4410@crunchy1[lab1]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                64
On-line CPU(s) list:   0-63
Thread(s) per core:    2
Core(s) per socket:    8
Socket(s):             4
NUMA node(s):          8
Vendor ID:             AuthenticAMD
CPU family:            21
Model:                 1
Model name:            AMD Opteron(TM) Processor 6272
Stepping:              2
CPU MHz:               2100.159
BogoMIPS:              4199.86
Virtualization:        AMD-V
L1d cache:             16K
L1i cache:             64K
L2 cache:              2048K
L3 cache:              6144K
NUMA node0 CPU(s):     0-7
NUMA node1 CPU(s):     8-15
NUMA node2 CPU(s):     32-39
NUMA node3 CPU(s):     40-47
NUMA node4 CPU(s):     48-55
NUMA node5 CPU(s):     56-63
NUMA node6 CPU(s):     16-23
NUMA node7 CPU(s):     24-31
```

number of cores = 4*8 = 32
number of threads: 4*8*2 = 64

a. We don't get a speedup or we get a lesser speedup when the number of processes converges towards the number of unknowns. For smaller problems (less unknowns) this decreases in speedup happens earlier whereas for larger problems more processes produce a speedup up until a certain point when speedup decreases. Speedup dramatically decreases when the number of processes is the same as the total number of cores in the system and decreases more when it's the same as the number of total threads.
b. We don't get a speedup as the number of processes converges towards the number of unknowns because of more calls to MPI_Allgatherv done by each process. Especially when the problem size is small the cost of the call to this function easily outweighs the benefits from more processes so speedup decreases early on. For problems of bigger size the benefit from parallelization are bigger so we get a speedup until the number of processes doing the call to the above function starts being too costly to benefit from each extra process. We also don't get a speedup when we ask the system to use all its resources towards our computations, that is when the number of processes is equal to the number of cores or threads.
c. We get a speedup when the number of processes is low especially relative to the problem size and relative to the total number of cores on the system.
d. When the number of processes is low there aren't many calls made to gather the information from every process so there isn't much time lost waiting for MPI_Allgatherv. Speedup is also pretty significant when the number of processes is about half as many as the total number of cores in the system, at this point the load is well balanced between

the processes and the system doesn’t have to allocate all of its resources towards our program only.