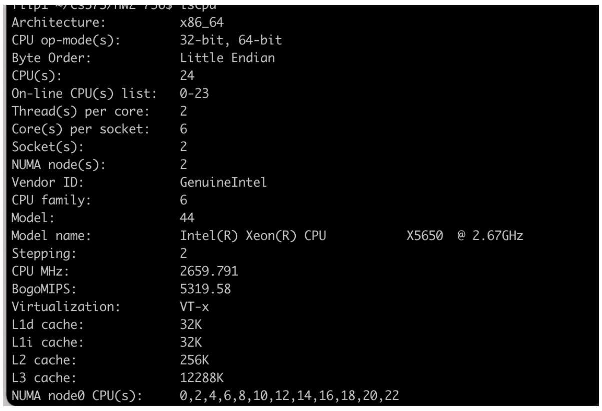
Project #1: Numeric Integration with OpenMP Reduction

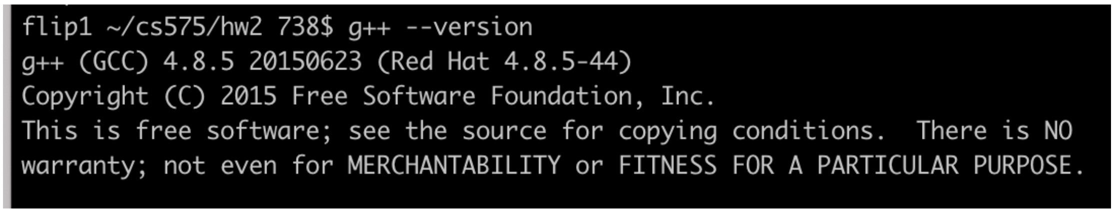
Danlin Song

songdan@oregonstate.edu

1. Tell what machine you ran this on
   I ran this project on OSU Linux server.

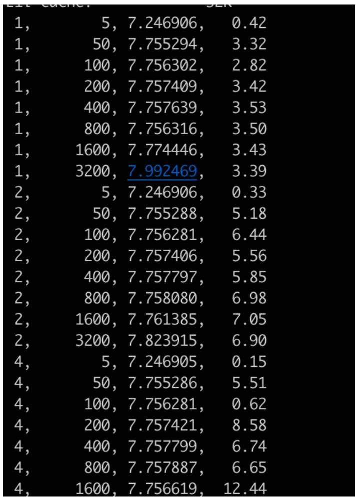


```
flip1 ~/cs575/hw2 736$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                24
On-line CPU(s) list:   0-23
Thread(s) per core:    2
Core(s) per socket:    6
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 44
Model name:            Intel(R) Xeon(R) CPU           X5650  @ 2.67GHz
Stepping:              2
CPU MHz:               2659.791
BogoMIPS:              5319.58
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              12288K
NUMA node0 CPU(s):     0,2,4,6,8,10,12,14,16,18,20,22
```



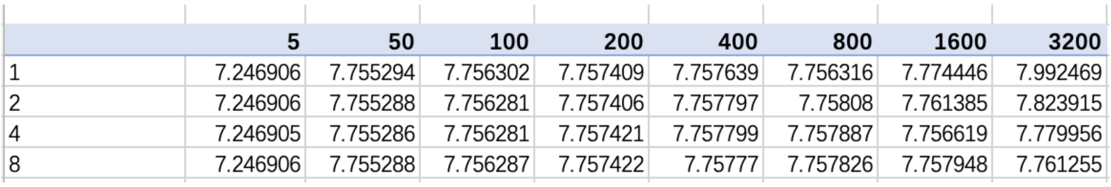
```
flip1 ~/cs575/hw2 738$ g++ --version
g++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44)
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Running results:



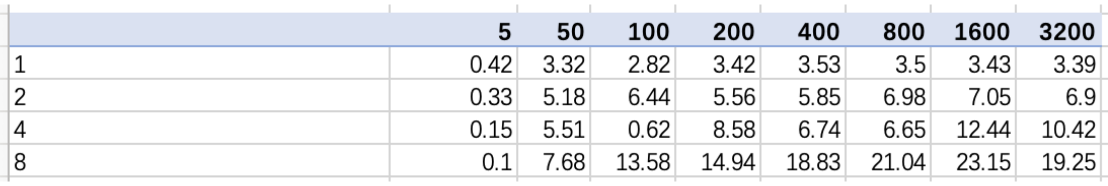
```
L1i cache:             32K
 1,       5, 7.246906,   0.42
 1,      50, 7.755294,   3.32
 1,     100, 7.756302,   2.82
 1,     200, 7.757409,   3.42
 1,     400, 7.757639,   3.53
 1,     800, 7.756316,   3.50
 1,    1600, 7.774446,   3.43
 1,    3200, 7.992469,   3.39
 2,       5, 7.246906,   0.33
 2,      50, 7.755288,   5.18
 2,     100, 7.756281,   6.44
 2,     200, 7.757406,   5.56
 2,     400, 7.757797,   5.85
 2,     800, 7.758080,   6.98
 2,    1600, 7.761385,   7.05
 2,    3200, 7.823915,   6.90
 4,       5, 7.246905,   0.15
 4,      50, 7.755286,   5.51
 4,     100, 7.756281,   0.62
 4,     200, 7.757421,   8.58
 4,     400, 7.757799,   6.74
 4,     800, 7.757887,   6.65
 4,    1600, 7.756619,  12.44
```

2. What do you think the actual volume is?
   I think the actual volume is 7.7.



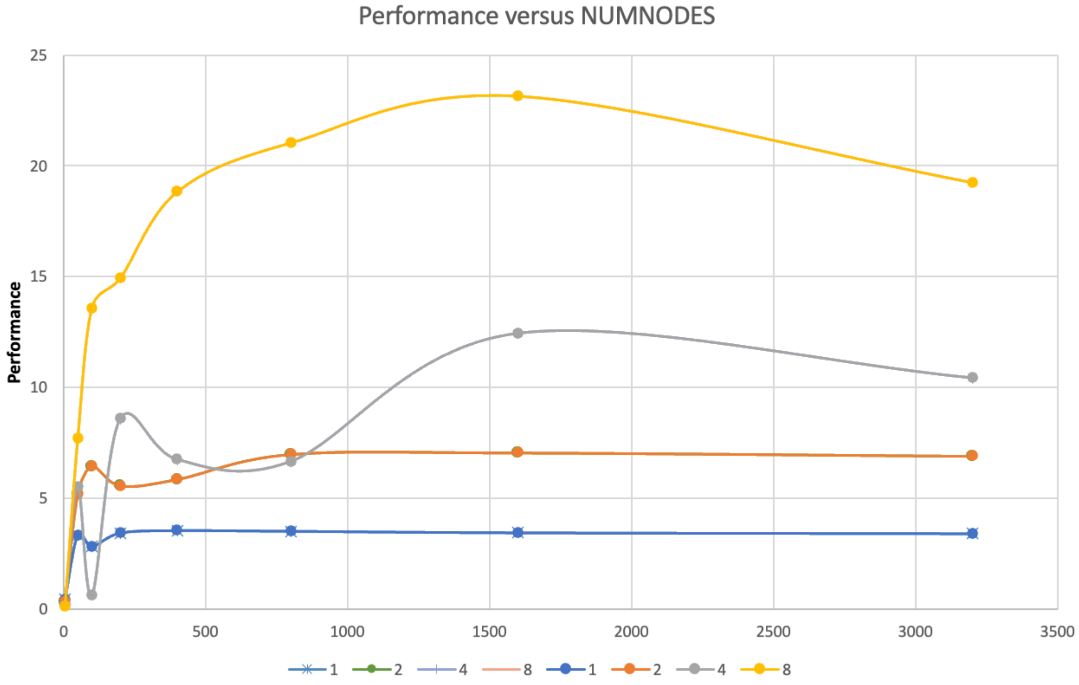
	5	50	100	200	400	800	1600	3200
1	7.246906	7.755294	7.756302	7.757409	7.757639	7.756316	7.774446	7.992469
2	7.246906	7.755288	7.756281	7.757406	7.757797	7.75808	7.761385	7.823915
4	7.246905	7.755286	7.756281	7.757421	7.757799	7.757887	7.756619	7.779956
8	7.246906	7.755288	7.756287	7.757422	7.75777	7.757826	7.757948	7.761255

3. Show the performances you achieved in tables and two graphs showing:
   Performance for different threads and nodes

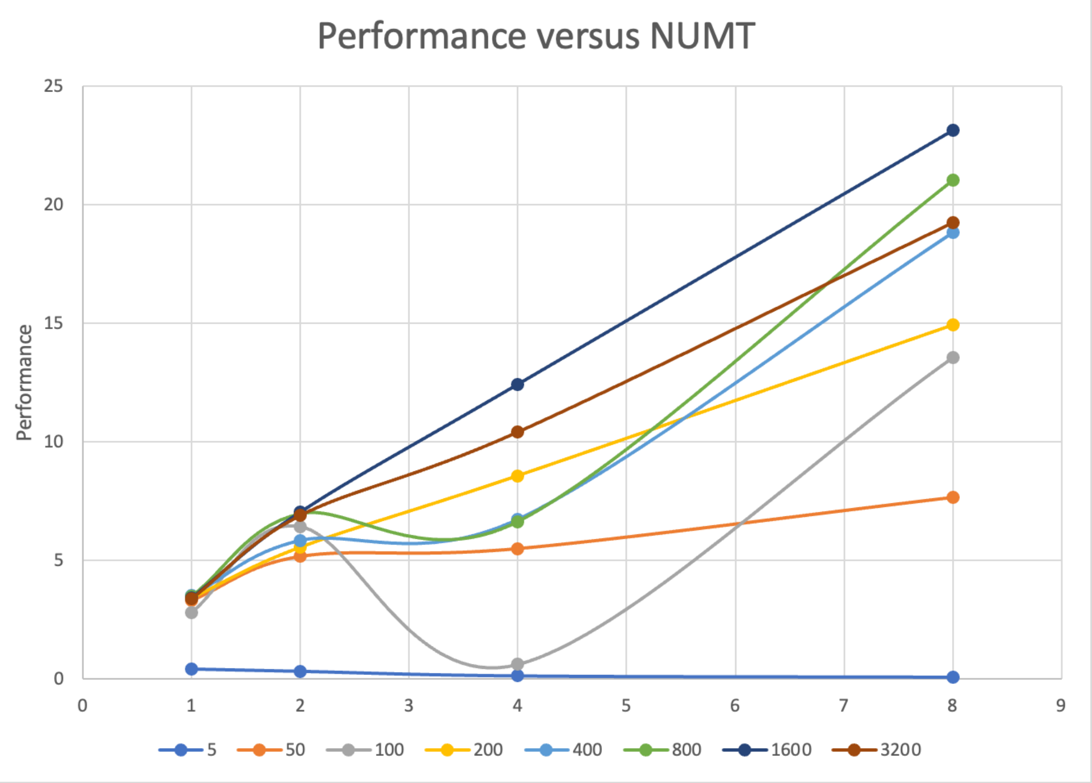


	5	50	100	200	400	800	1600	3200
1	0.42	3.32	2.82	3.42	3.53	3.5	3.43	3.39
2	0.33	5.18	6.44	5.56	5.85	6.98	7.05	6.9
4	0.15	5.51	0.62	8.58	6.74	6.65	12.44	10.42
8	0.1	7.68	13.58	14.94	18.83	21.04	23.15	19.25

A. Performance as a function of NUMNODES with colored lines showing different NUMT values



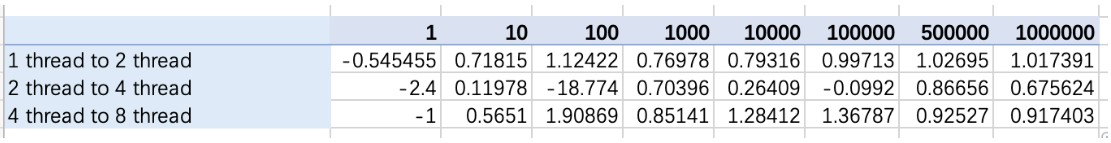
B. Performance as a function of NUMT with colored lines showing different NUMNODES values



4. What patterns are you seeing in the speeds?
Performance becomes convergent when running with 1 or 2 threads and NUMNODES less than 5. When running on more than four threads, the performance increases to the highest point then decreases as the number of

NUMNODES increases and outperforms single threads. When NUMNODES is greater than 50, the performance becomes good as the number of threads increases.

5. Why do you think it is behaving this way?
   When the NUMNODES and NUMT numbers are small, the number of calculations is too small, which makes the results not accurate enough. When NUMNODES and NUMT are getting larger, more and more calculations are made, then the volume values and performance converge and become more and more accurate.
6. What is the Parallel Fraction for this application, using the Inverse Amdahl equation?



	1	10	100	1000	10000	100000	500000	1000000
1 thread to 2 thread	-0.545455	0.71815	1.12422	0.76978	0.79316	0.99713	1.02695	1.017391
2 thread to 4 thread	-2.4	0.11978	-18.774	0.70396	0.26409	-0.0992	0.86656	0.675624
4 thread to 8 thread	-1	0.5651	1.90869	0.85141	1.28412	1.36787	0.92527	0.917403

   4-to-8-thread Fp = (8/(8-4))*(1-(1/(23.15/12.44)) = 0.9253
7. Given that Parallel Fraction, what is the maximum speed-up you could ever get?
   Max speedup = 1/1-Fp = 1/1-0.9253= 13.37