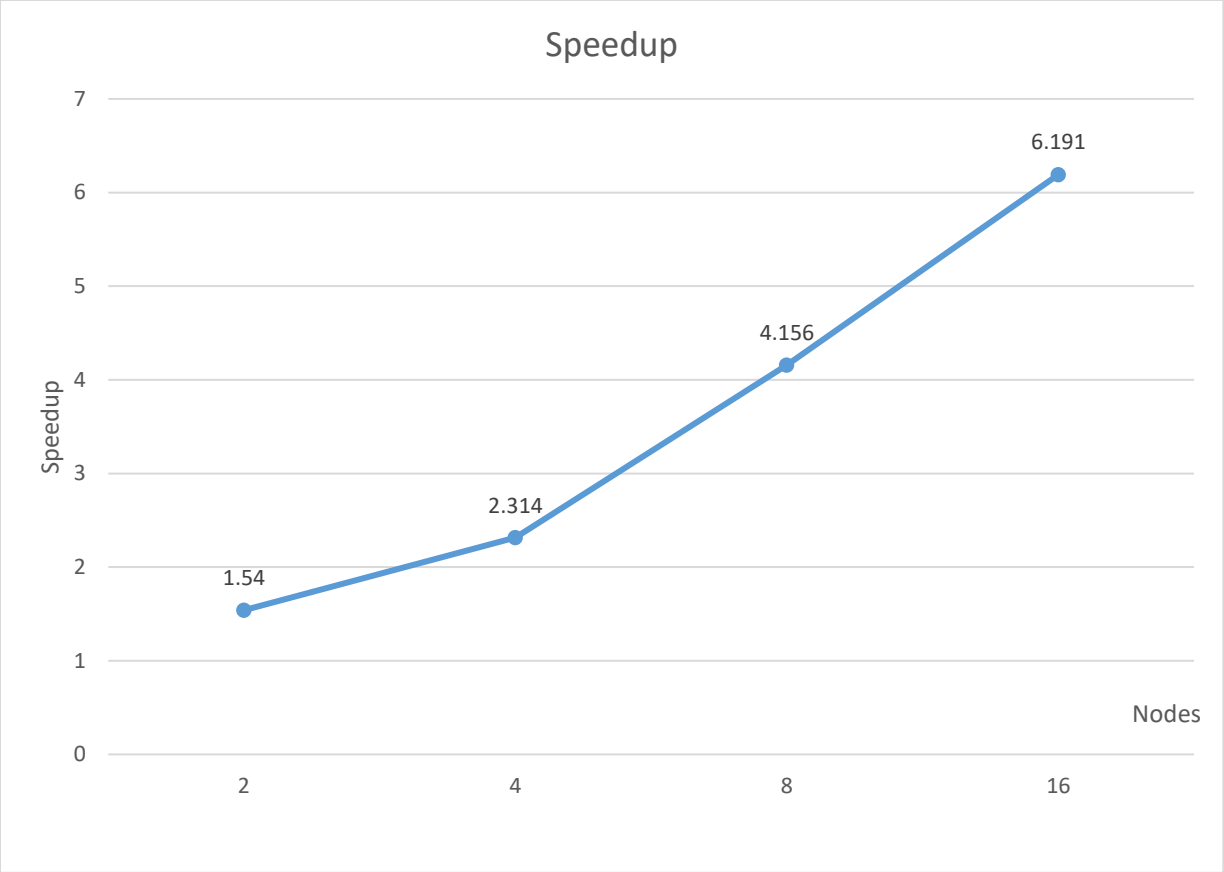
ECE 563
Songrui Li
0025338817
li884@purdue.edu
2018/4/27

ECE 563 Small Project Step 3 Report

I. Matrix Multiply with Cannon Algorithm

a) Speed Up

NODES	SEQUENTIAL	2	4	8	16
1ST RUN	2.634	1.656	1.051	0.678	0.423
2ND RUN	2.642	1.746	1.164	0.535	0.365
3RD RUN	2.616	1.723	1.197	0.686	0.486
MEAN	2.631	1.708	1.137	0.633	0.425
SPEEDUP	N/A	1.540	2.314	4.156	6.191



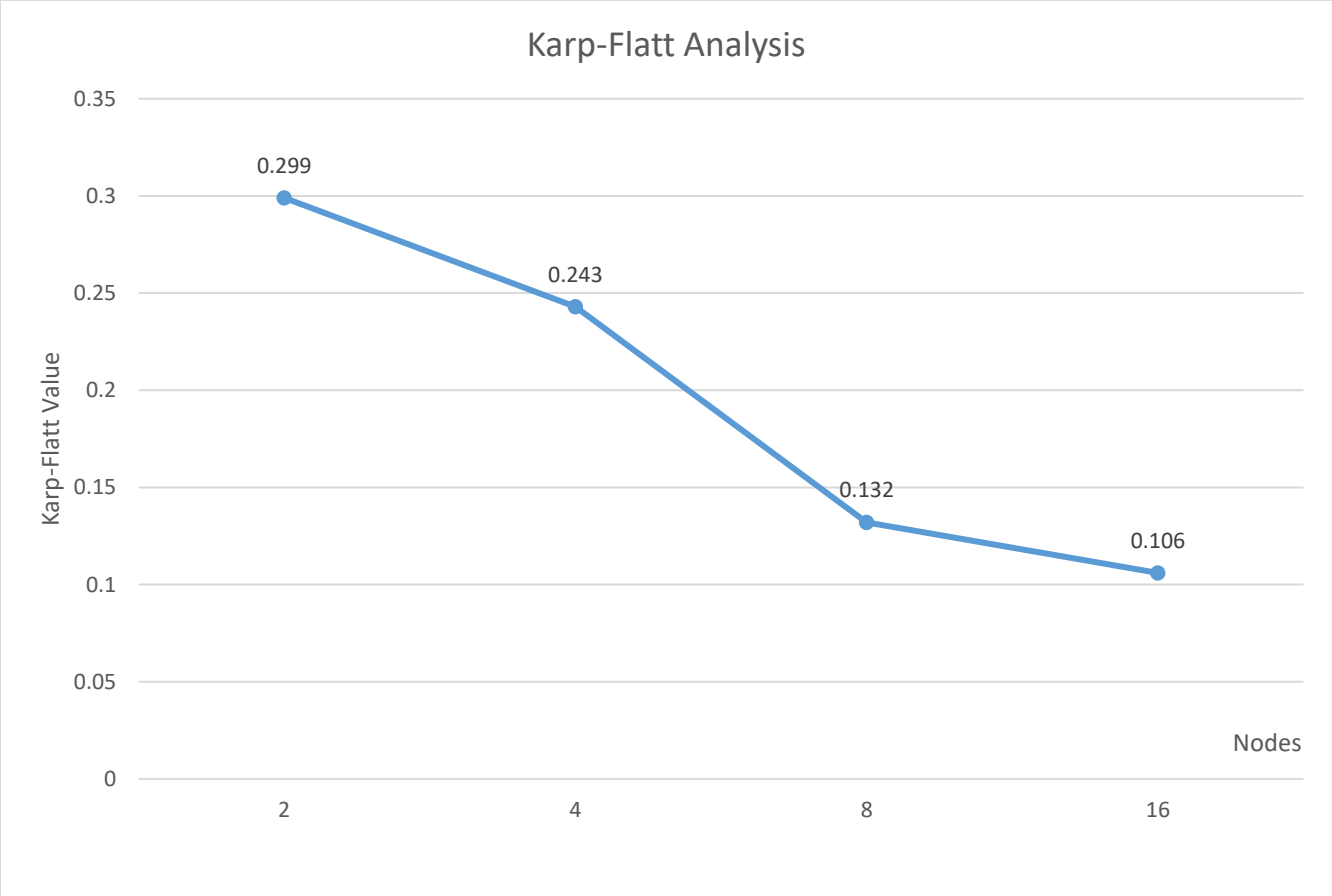
The speedup increases when there are more nodes. The master thread sends the data of the matrix to other work threads. After worker threads compute the related rows and columns, it sends the data back to the master thread. Each node receives same amount of workload. Therefore, the speedup increases with more nodes.

b) Karp-Flatt Analysis

NODES	SEQUENTIAL	2	4	8	16
KARP-FLATT	N/A	0.299	0.243	0.132	0.106

Karp-Flatt Metric calculation was done as follows:

$$e = \frac{\left(\frac{1}{\psi} - \frac{1}{P}\right)}{\left(1 - \frac{1}{P}\right)}$$



The parallelization is better when the Karp-Flatt value is smaller. According to the data result, the value decreases with the number of nodes become larger. So, the parallelization becomes better.

c) Iso-efficiency Analysis

NODES	SEQUENTIAL	2	4	8	16
MEAN RUN TIME	2.631	1.708	1.137	0.633	0.425
ISO-EFFICIENCY	N/A	0.770	0.579	0.519	0.387
OVERHEAD	N/A	0.785	1.917	2.433	4.169

Parallel overhead T_0 = Nodes Number x $T_p - T_1$

Iso_efficiency = $\frac{1}{(1+\frac{T_0}{T_1})}$

General Equations:

$$T_O = Total\ Overhead$$

$$T_1 = Single\ Processor\ Time$$

$$= W * t_C = (Units\ of\ Work) * (Time\ per\ unit\ of\ work)$$

$$T_P = Time\ Parallel$$

$$PT_P = T_1 + T_O = Total\ Time$$

$$S = \frac{T_1}{T_P} = \frac{PT_P}{T_1 + T_O} = Speedup$$

$$T_P = Time\ to\ Receive\ Data +$$

$$Time\ to\ Multiply\ Matrix\ Segment +$$

$$Time\ to\ Share\ Matrix * Sub\ Matricies$$

$$Time\ to\ Return\ Results$$

$$T_P = (\text{Items to Receive}) * (\text{Cores Receiving}) * (\text{Comm Time}) +$$

$$(\text{n multiplications} + (\text{n} - 1)\text{additions}) * \left(\frac{n^2}{\text{P}}\ \text{entries}\right) +$$

$$(\text{Items to Share}) * 2 * (\text{Cores Sharing}) * (\text{Comm Time}) +$$

$$(\text{Number of Communications}) * (\text{Data Returned}) * (\text{Time Return})$$

$$T_p = \frac{n^2}{P} * P * t_C +$$

$$(n * t_{mult} + (n-1) * t_{add}) * \left(\frac{n^2}{P}\right) +$$

$$\frac{n}{P} * 2 * P * t_C +$$

$$\log_2 P * \frac{n}{P} * t_C$$

$$T_P = (n^2 + 2 * n + \frac{n}{P} * \log_2 P) * t_c + (n * t_{mult} + (n-1) * t_{add}) * \frac{n^2}{P}$$

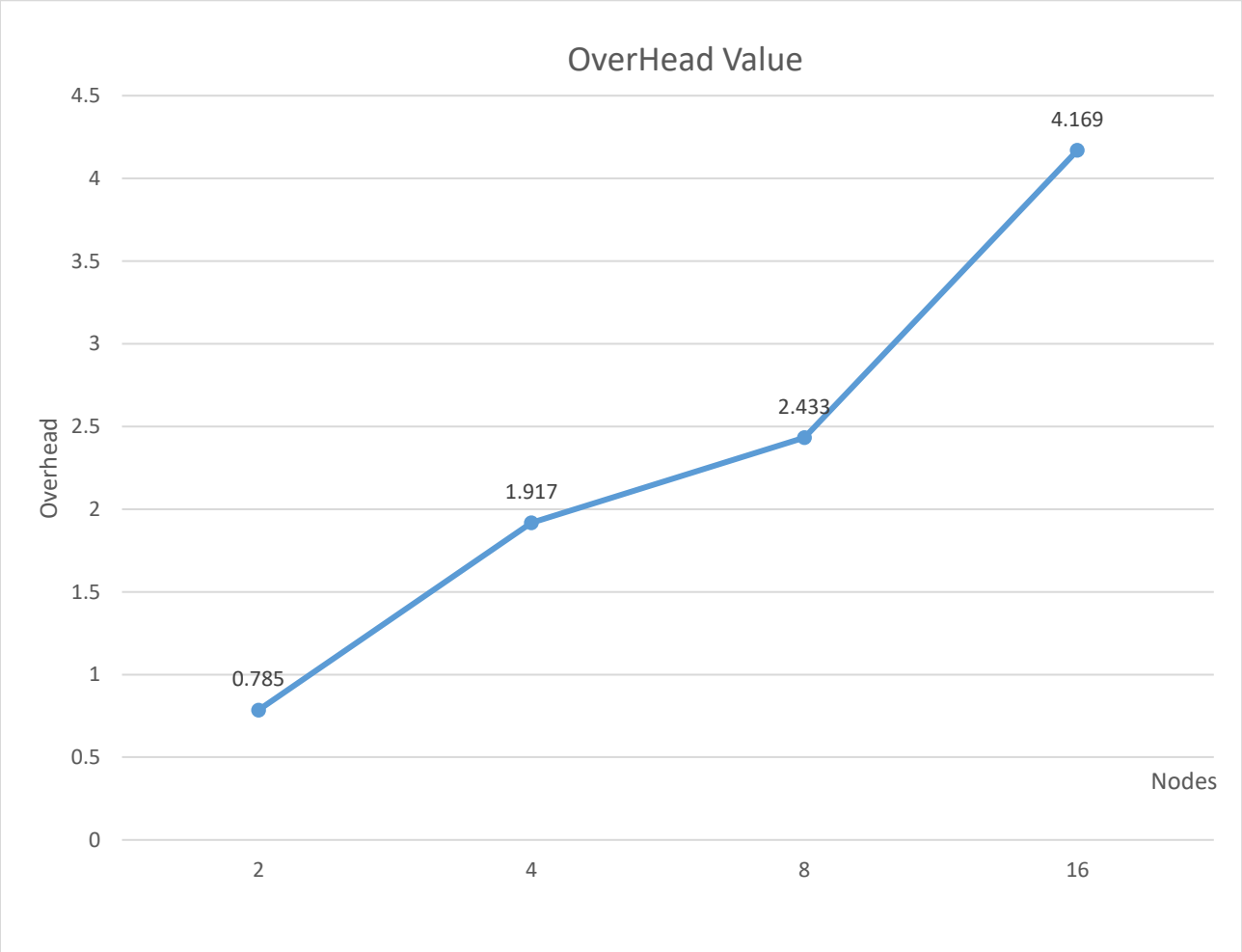
$$PT_P = T_0 + T_1$$

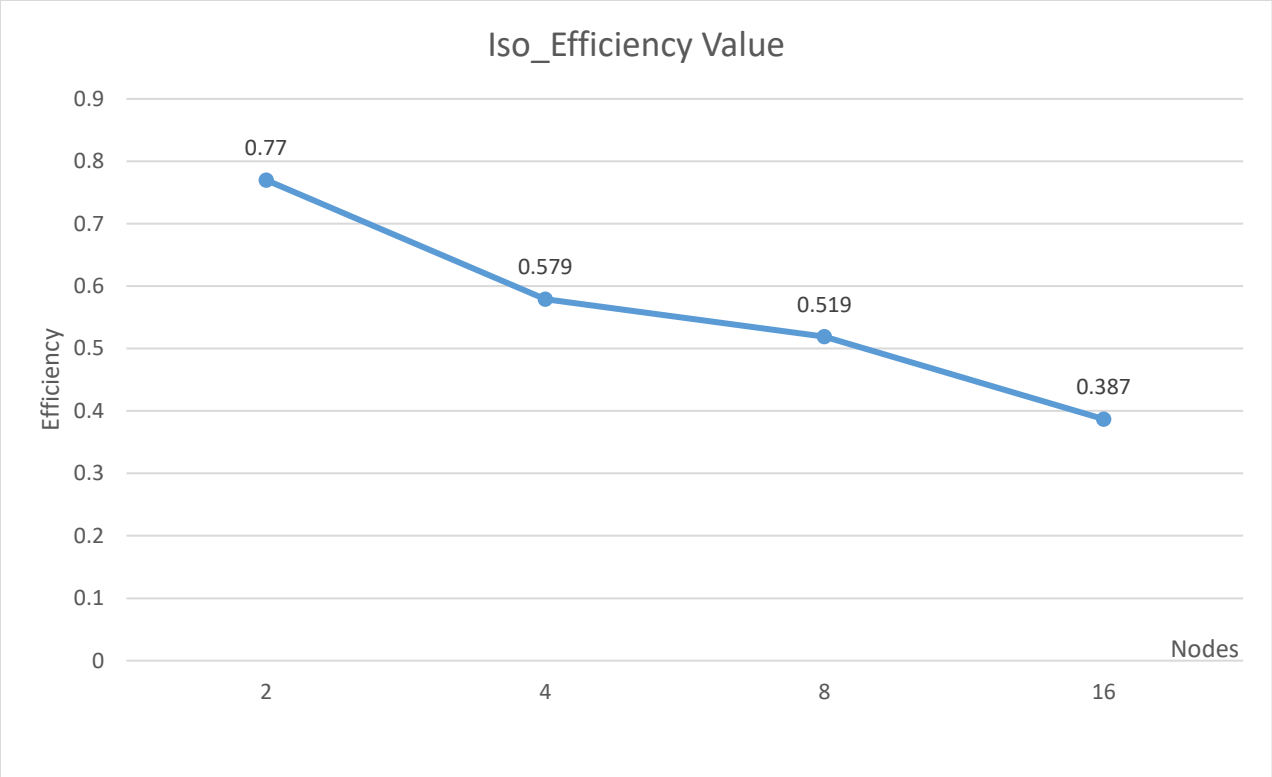
$$T_O = (n^2 + 2 * n + \frac{n}{P} * \log_2 P) * t_c * P$$

$$T_1 = (n * t_{mult} + (n-1) * t_{add}) * n^2$$

$$\theta(P * \log_2 P) = Iso - Efficiency$$

When determining the final iso-efficiency of this algorithm, the main concern was how cost would scale as the number of processors increased. When looking at the actual algorithm time, the computations were divided in such a way that the cost of multiplying the actual matrices was split evenly, so that cost does not increase due to parallelization. The cost that does increase is the cost of sharing the data between cores, which is described by the final reduction necessary. The initial distribution is linear, so even though more communications would be needed as the number of cores increased, the time of each communication would decrease.



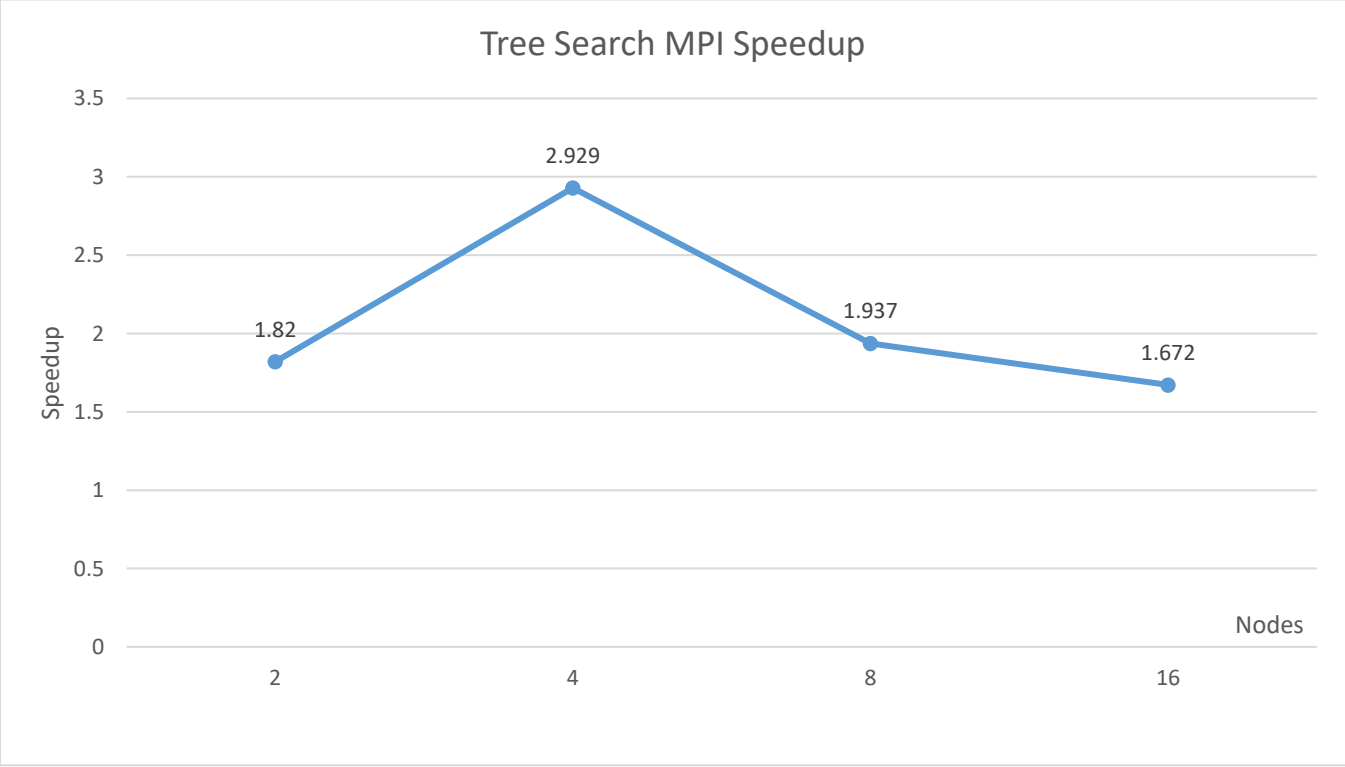


The overhead increases when number of nodes become larger and the iso_efficiency decreases at the same time. Therefore, there are more overhead due to the parallelization and less iso_efficiency when the number of nodes increase.

II. Tree Search MPI Version

a) Speed Up

NODES	SEQUENTIAL	2	4	8	16
1ST RUN	0.0325	0.0363	0.0076	0.020	0.0167
2ND RUN	0.0376	0.0215	0.0238	0.0132	0.0326
3RD RUN	0.0485	0.0074	0.0091	0.0280	0.0216
MEAN	0.0395	0.0217	0.0135	0.0204	0.0236
SPEEDUP	N/A	1.820	2.929	1.937	1.672



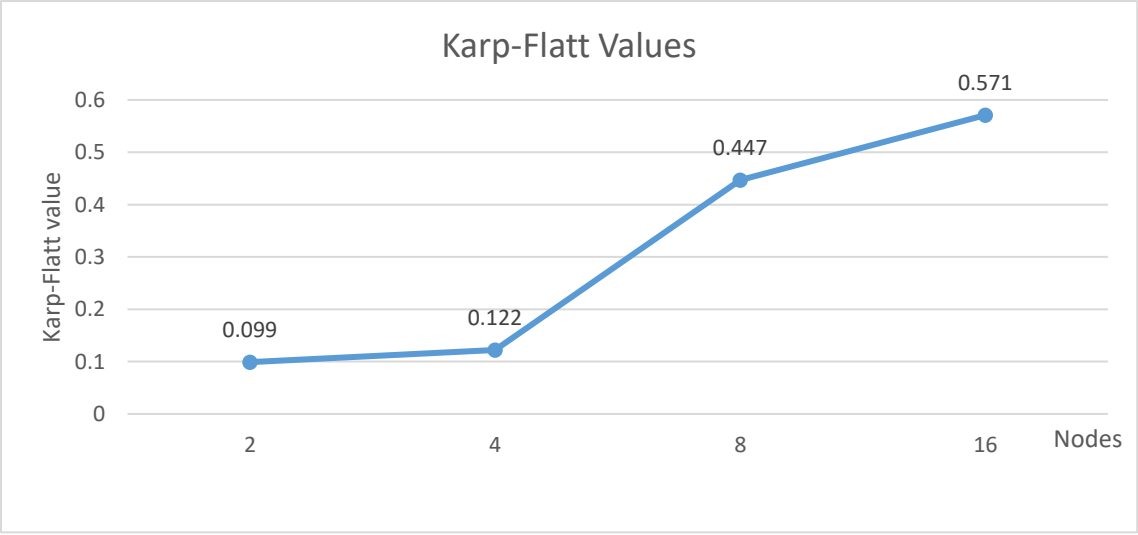
The master thread oversees allocate different part of the tree to sub threads and sub threads will do the search algorithm. The speedup becomes largest when there are 4 nodes and then decreases with larger nodes.

b) Karp-Flatt Analysis

NODES	SEQUENTIAL	2	4	8	16
KARP-FLATT	N/A	0.099	0.122	0.447	0.571

Karp-Flatt Metric calculation was done as follows:

$$e = \frac{\left(\frac{1}{\psi} - \frac{1}{P}\right)}{\left(1 - \frac{1}{P}\right)}$$



According to the figure above and Karp-Flatt theory, the efficiency of the program is increasing with the number of threads increasing.

c) Iso-efficiency Analysis

NODES	SEQUENTIAL	2	4	8	16
MEAN RUN TIME	0.0395	0.0217	0.0135	0.0204	0.0236
ISO-EFFICIENCY	N/A	0.910	0.732	0.242	0.105
OVERHEAD	N/A	0.0040	0.0144	0.124	0.339

Parallel overhead T_0 = Nodes Number x $T_p - T_1$

Iso_efficiency = $\frac{1}{(1+\frac{T_0}{T_1})}$

General Equations:

$$T_O = Total\ Overhead$$

$$T_1 = Single\ Processor\ Time$$

$$= W * t_C = (Units\ of\ Work) * (Time\ per\ unit\ of\ work)$$

$$T_P = Time\ Parallel$$

$$PT_P = T_1 + T_O = Total\ Time$$

$$S = \frac{T_1}{T_P} = \frac{PT_P}{T_1 + T_O} = Speedup$$

$$T_P = Time\ to\ Scan\ Subset\ of\ Files + Time\ to\ Return\ Results$$

$$T_P = \frac{file\ size}{P} * t_{parse} + \log_2 P$$

$$P * T_P = T_O + T_1$$

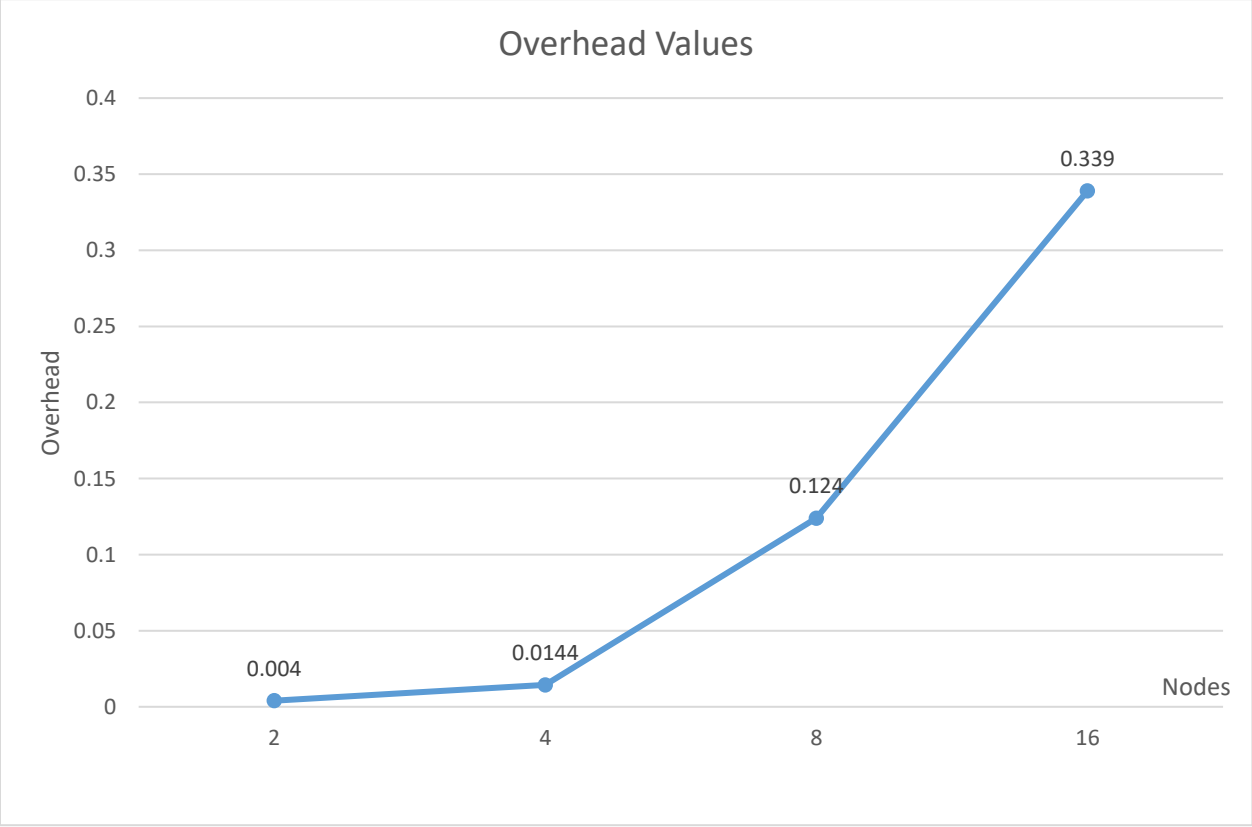
$$(file\ size) * t_{parse} + P * \log_2 P = T_O + T_1$$

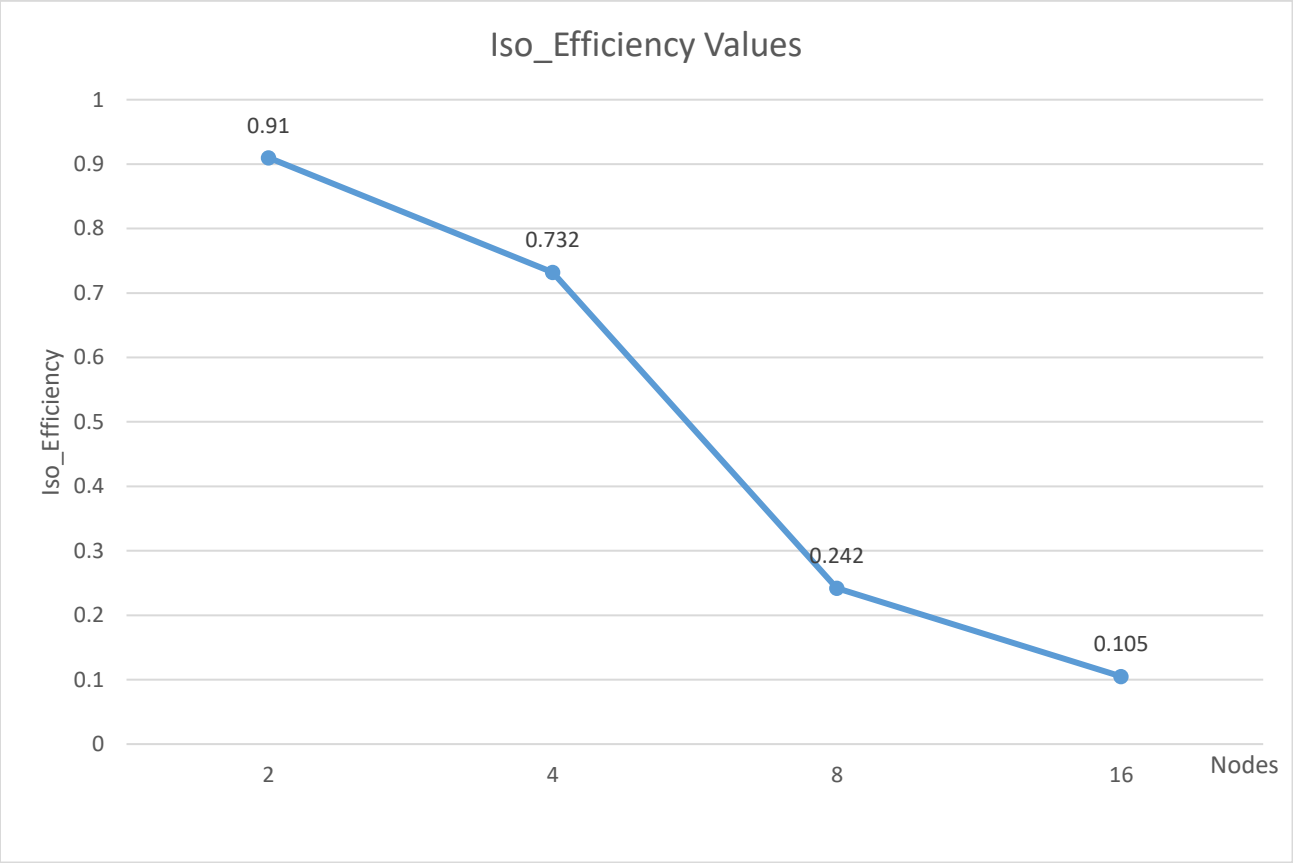
$$T_O = P * \log_2 P$$

$$T_1 = (file\ size) * t_{parse}$$

$$\theta(P * \log_2 P) = Iso - Efficiency$$

As shown above, the Iso-Efficiency of the parallelized binary tree search depends solely on the number of threads used and is based on the cost of the reduction at the end of the search.





According to the figure above, the overhead of the program increasing significantly with the increasing of number of threads. So, the efficiency of the program drops.