International Journal of General Engineering and Technology (IJGET)
ISSN(P): 2278-9928; ISSN(E): 2278-9936
Vol. 6, Issue4, Jun – Jul 2017; 55-62
© IASET

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

PERFORMANCE EVALUATION OF ALL PAIR SHORTEST PATH PARALLEL ALGORITHM USED IN BIG DATA

M. CHITHIK RAJA[1] & M. MUNIR AHMED RABBANI[2]

[1]Department of IT, AMET University, Kanathur, Chennai, Tamil Nadu, India

[2]Department of Computer Application, B.S.Abdur Rahman University, Chennai, Tamil Nadu, India

ABSTRACT

In this paper, we have contrasted with assessing four diverse parallel calculations, for the all-set's most limited way issue, by utilizing execution models. An important issue is the huge information correspondence arranges innovation, transportation and hardware issues. We have broken down four distinctive parallel calculations, which are utilized as a part of the huge information correspondence arrange. This paper demonstrates that, three of the four calculations can be finest in various circumstances, contingent upon exchanges between enormous information on big data computation and communication costs.

KEYWORDS: Floyds, Dijkstra, Parallel, Interception Point, Boundaries, Node, Big Data

INTRODUCTION

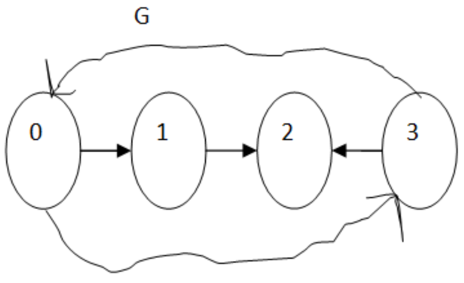
The calculations, to locate the short range remove from a source capture attempt direct S, towards an objective block attempt point T, is in associated organizes. Things being what they are, the most incredible calculations for this issue genuinely locate the short range remove from S to each conceivable target capture attempt point T, by building a briefest way tree. The most limited way tree indicates two snippets of data, for every hub v in the system: Dist (v) is the length of the briefest way (assuming any) from S to v; node (v) is the second-to-last interference point (assuming any) the briefest way (assuming any) from s to v. In this paper, we need to sum up the briefest way issue considerably, advance in the all set of short or run separate issue; we need to locate the most limited way from each conceivable source to each conceivable goal. In particular, for each combine of interference point u and v, we have to work out the accompanying altogether: dist (u, v) is the length of the most limited way (assuming any), from u to v; node (u, v) is the second-to-last block attempt point (assuming any) on the briefest way (assuming any), from u to v. For instance, for any block attempt point v, we have dist (v, v) = 0 and node (v, v) = Null. On the off chance that, the most limited way from u to v is just a single limit long, at that point dist(u, v) = w(uv) and node(u, v) = u. In the event that, there is no most limited way, from u to v both on the grounds that, there's no way by any stretch of the imagination, or on the grounds that, there's a negative cycle, then dist (u, v) = ∞ and node (v, v) = Null. The yield of our most brief way calculations will be, a couple of V × V exhibits, including all V2separations and ancestor [1].

ALGORITHM EXPERIMENT

The all-pairs shortest-path problem involves, finding the shortest path between all pairs of interception points in a network. A network G=(V,E) comprises a set V, of N interception points,{v_i}, and a set E ≤ V X V of boundaries connecting interception points, in V. In a directed network, each boundary also has a direction, so boundaries {v_i, vj} and (v_i, vj), where i is not equal to j, are different.

A network can be represented as an adjacency matrix A, in which each element (i, j) represents the boundary between element i and j A_{ij}=1. if there is a boundary (vi,vj) ; otherwise, A_{ij} =0[2]

A path from interception point vi to interception point vj is a sequence of boundaries (v_i,v_k), (v_k,v_t) … (v,v_j) from E, in which no interception point appears more than once. For example, (1, 3), (3, 0), is a path from interception point 1 to interception point 0, in Figure 1. The shortest path between two interceptions points vi and vj in a network, is the path that has the fewest boundaries. The single-source shortest-path problem requires finding the shortest path from a single interception point to all other interception points, in a network. The all-pairs shortest-path problem requires finding the shortest path between, all pairs of interception points in a network. We consider the second problem and present four different parallel algorithms, two based on a sequential shortest-path algorithm due to Floyd and two based on a sequential algorithm, due to Dijkstra. All four algorithms take as input as, N x N adjacency matrix A and compute an N x N matrix S, with $S_{ij,}$ the length of the shortest path from vi to v_j, or a distinguished value infinitive, if there is no path.



$$A = \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 1 & 0 \end{array}$$

Figure 1: A Simple Directed Network, G, and Its Adjacency Matrix, A

Floyd's Algorithm

```
Procedure sequential Floyd
begin
$I_{ij}(0) = 0$ if $i=j$
$I_{ij}(0)=$ length((vi,vj)) if boundary exists and $i \neq j$
$I_{ij}(00= \infty$ otherwise
for k = 0 to N-1
for i =0 to N-1
for j = 0 to N-1
$I_{ij}(k+1) = \min(I_{ij}(k), I_{ik}(k)+I_{kj}(k))$
endfor
endfor
endfor
S=I(N)
end
```

Algorithm 1: Floyds all Pairs Shortest Path Algorithm

Floyd's all-sets, most limited way calculation is given as an Algorithm-1. It determines the grid S in N steps, building at each progression k, a halfway lattice $I_{(k)}$ containing the best-known most limited separation between each combine of hubs. At first, each $I_{ij(0)}$ is set to the length of the boundary, if the boundary exists and to something else. The kth venture of the calculation considers each I_{ij} thus and decides, if the best-known way from v_i to v_j is longer than the joined lengths of the best-known ways, from v_i to v_j and from v_k to v_j Provided that, this is true that the passage Iij is refreshed to mirror the shorter path[3]. This correlation operation is played out a sum of N3 times; consequently, we can rough the successive cost of this calculation as t_cN3, where t_c is the cost of a solitary examination operation.

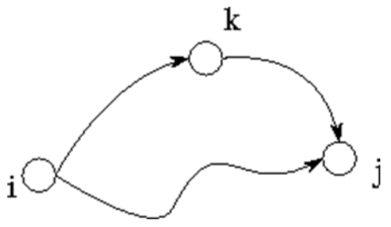


Figure 2: The Fundamental Operation in Floyd's Sequential Shortest-Path Algorithm: Determine Whether a Path Going from V_i to V_j Via V_k is Shorter than the Best-Known Path from V_i to V_j

Parallel Floyd 1

The principal parallel Floyd calculation depends on a one-dimensional, push confidence zone breakdown of the middle network-1 and the yield framework S. Notice that, this implies that the calculation can use at most N processors. Each assignment has at least one nearby columns of I and is in charge of performing calculation, on those lines. That is, it executes the accompanying rationale.

```
for $k = 0$ to $N-1$
for $I$ = local_i_start to local_i_end
for $j = 0$ to $N-1$
$I_{ij}(k+1) = \min(I_{ij}(k), I_{ik}(k)+I_{kj}(k))$
endfor
endfor
endfor
```

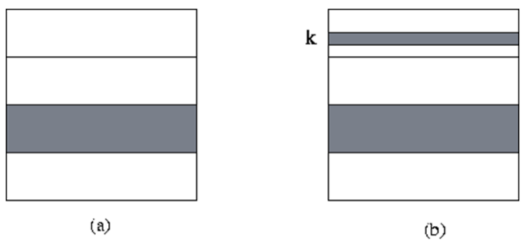


Figure 3: Parallel Version of Floyd's Algorithm based on a one-Dimensional Decomposition of the I Matrix. In (a), the Data Allocated to a Single Task are Shaded: A Contiguous Block of Rows. In (b), the Data required by this Task in the kth Step of the Algorithm are Shaded: Its Own Block and the kth Row

In the kth step, each undertaking requires, not withstanding its nearby information, the qualities I_{k0}, $I_{k1, \ldots,}$ I_{kN-1} that is, the kth line of I, (Figure3). Henceforth, we determine that, the undertaking with this column communicate it to all other tasks [4]. This correspondence can be performed, by utilizing a tree structure in log P steps. Since there are N such communicates and each message has measure N, the cost is

$$T_{Floyd1} = t_c \, N^3 \big/ P + \log P\left(t_c + t_w N\right)$$

Parallel Floyd 2

An alternative parallel version of Floyd's algorithm uses, a two-dimensional decomposition of the various matrices. This version allows the use of up to N^2 processors and requires that, each task executes the following logic[5].

```
for $k = 0$ to $N-1$
for $i$ = local_i_start to local_i_end
for $j$ = local_j_start to local_j_end
$I_{ij}(k+1) = \min(I_{ij}(k), I_{ik}(k)+I_{kj}(k)$
endfor
endfor
endfor
```

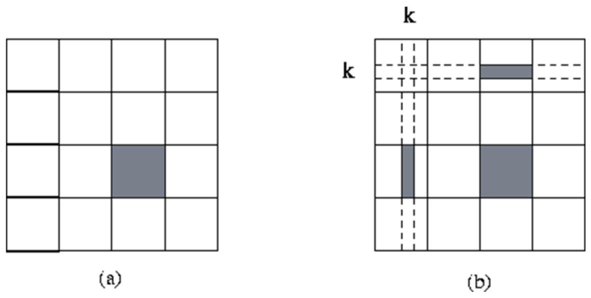


Figure 4: Parallel Version of Floyd's Algorithm Based on a two-Dimensional Decomposition of the I Matrix. in (a), the Data Allocated to a Single Task are Shaded: A Contiguous Submatrix. In (b), the Data required by this Task in the k th Step of the Algorithm are Shaded: Its Own Block, and Part of the k Th Row and Column

In each progression, the undertaking requires, not withstanding its neighborhood information, N/√p values from two assignments situated in a similar line and segment of the 2-D errand exhibit, (Figure4). Consequently, correspondence necessities at the kth step, can be organized as two communicate operations: from the assignment in each line, that has some portion of section k to every single other errand in that line and from the undertaking in every segment, that has some portion of line k to every single other undertaking in that segment. In each of N steps, N/√p values must be communicated to the √p undertakings, in each line and segment and the aggregate cost is

$$T_{Flod2} = t_c + 2N \log \sqrt{P}\left(t_c + t_w \, N/\sqrt{P}\right)$$
$$= t_c \, N^3 \big/ P + N \log P\left(t_c + t_w \, N \big/ \sqrt{P}\right)$$

Dijkstra's Algorithm

Dijkstra's single-source most limited way calculation, figures every single briefest way from a solitary capture attempt point, i.e. it can likewise be utilized for the all-sets most brief way issue, by the basic catalyst of applying it N times, once to every interference point $v_0,..., v_{n-1}$. Dijkstra's successive single-source calculation is given as Algorithm2. It keeps up as T, the arrangement of block attempt focuses, for which most limited ways have not been found and as d_i is the briefest known way, from vi to interference point v_j. At first, T=V and all $d_i=\infty$. At each progression of the calculation, the capture attempt point v_m in T, with the littlest d esteem, is expelled from T[6]. Each neighbor of v_m in T is inspected to see, whether a way through v_m would be shorter than the, as of now best-known way (Figure5).

```
Procedure sequential_dijikstra
Begin
$d_s= 0$
$d_i= \infty$, for $i \neq s$
T= V
for i= 0 to N-1
find $v_m$ € T with minimum $d_m$
for each boundary $(v_m,v_i)$ with $v_i$ € T
if $(d_i > d_m + length((v_m, v_i)))$ then $d_t = d_m + length((v_m, v_i))$
endfor
$T = T - v_m$
endfor
End
```

Algorithm 2: Single Source Dijikstra Algorithm

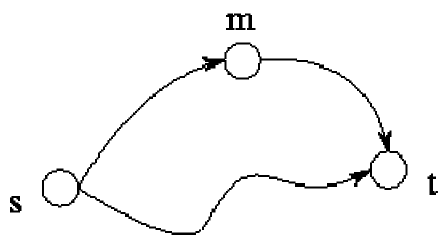


Figure 5: The Contrast Operation Performed in Dijkstra's Single-Source Shortest-Path Algorithm. The Finest-Known Path from the Source Interception Point V_s to Interception Point V_t is Compared with the Path that Leads from V_s to V_m and Then to V_t

All-pairs shortest path algorithm executes Algorithm2 *N* times, once for each interception point [7]. This involves $O(N^3)$ comparisons and takes time as $N^3 t_a F$, where t_a is the cost of a single comparison in Floyd's algorithm and *F* is a constant. Empirical studies show that, $F \approx 1.6$; that is, Dijkstra's algorithm is slightly more expensive than Floyd's algorithm.

Parallel Dijkstra 1

The first parallel Dijkstra algorithm, replicates the network in each of P tasks. Each task executes the sequential algorithm, for *N divided by P* interception points [8]. This algorithm requires no communication, but can utilize at most N processors. Because the sequential Dijkstra algorithm, is F times slower than the sequential Floyd algorithm, the parallel algorithm's execution time is

$$T_{Dijk1} = t_c \; F \, N^3/P$$

Parallel Dijkstra 2

The second parallel Dijkstra algorithm, allows for the case when P>N. We label N lay down of P, divided N responsibilities [9]. Each lay down of responsibilities is given the entire network and is responsible for computing short range paths, for a single interception point (Figure5). Within each lay down of responsibilities, the interception points of the networks are partitioned. Therefore, the process Find v_m€ t is with minimum d_m. First, a local calculation is used to find the local interception point, with minimum d and second, a decline linking all P/N responsibilities in the same lay down, in order to conclude the globally minimum d_m[10][11]. The cutback can achieve by means of the butterfly communication structure, in log P divided by N steps. Hence, as the cutback is performed N times and involves two values, the total cost of this algorithm is

$$T_{Dijk2} = t_c \; F \, N^3/P \quad + N \log P/N \left(t_c + 2t_w\right)$$

- With adjacency matrix illustration, Floyd's algorithm has a worst case convolution of $O(n^3)$ where, n is the number of interception points
- If Dijkstra's algorithm is used for the same purpose, then with an adjacency list illustration, the worst case convolution will be $O(n_e \log n)$. Thus, if e is $O(n^2)$, then the convolution will be $O(n^3 \log n)$ while, if e is $O(n)$, then, the convolution is $O(n^2 \log n)$.

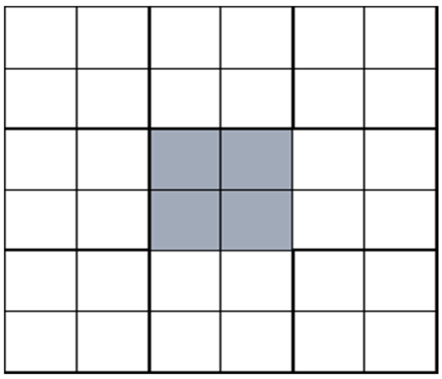
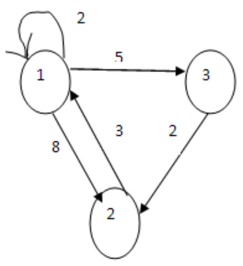


Figure 6: The Second Parallel Dijkstra Algorithm Allocates P/N Tasks to Each of N Instantiations of Dijkstra's Single-Source Shortest-Path Algorithm. in This Figure, N=9 and P=36, and One Set of P/N=4 Tasks is Shaded

ALGORITHM ANALYSIS AND PERFORMANCE EVALUATION RESULT



$$C = \begin{bmatrix} 2 & 8 & 5 \\ 3 & \infty & \infty \\ \infty & 2 & \infty \end{bmatrix}$$

$$A_0 = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{bmatrix} \quad A_1 = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ \infty & 2 & 0 \end{bmatrix} \quad A2 = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix} \quad A_3 = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix}$$

FINDINGS AND SUMMARY

Table 1, abridges the introduced models produced for the four, all-pairs shortest-path calculations. Plainly, Floyd 2 will dependably be more effective than Floyd 1. Both calculations have a similar calculation cost and send a similar number of messages, however, Floyd 2 convey impressively less information. Then again, Floyd 1 is less demanding to actualize. Calculations Dijkstra 1 and 2 will be more productive than Floyd 2, in specific conditions. For instance, Dijkstra 1 is more productive than Floyd 2, if $P \leq N$ and

Notwithstanding these variables, we should consider the way that calculations Dijkstra 1 and Dijkstra 2 rehash the system, P and P/N times, individually. This impersonation, may trade off the adaptability of these calculations. Additionally, the cost of reproducing an initially appropriated organize must be considered, if the most limited way calculation frames some portion of a bigger program in which, the system is spoken to as a circulated number structure.

$T_c(F\text{-}1)\, N^3/p < t_s N \log P + t_w N^2 \log P/\sqrt{P}$

Table 1: Performance of Four Parallel Shortest-Path Algorithms

Algorithm	t_c	t_s	t_w	Maximum P
Floyd 1	N^3 / P	N log P	N^2 log P	N
Floyd 2	N^3 / P	N log P	N^2 log P / √P	N^2
Dijkstra 1	N^3 F / P	0	0	N
Dijikstra 2	N^3 F / P	N log (P/N)	2N log (P/N)	N^2

CONCLUSIONS

We have examined the numerical execution models of parallel calculation, utilized as a part of enormous information that portray the execution time, effectiveness and adaptability of a parallel calculation, utilized as a part of huge information and correspondence parameters. We have additionally perceived, how these models can be utilized all through the parallel program plan and execution cycle:

Early in the planning procedure, we had described the calculation and correspondence prerequisites, of our parallel calculations, by building basic execution models. These models can be utilized, to pick between algorithmic choices, to recognize issue regions in the outline and to check that, calculations meet execution prerequisites.

Later, we refined our execution models and direct, straightforward examinations to decide obscure parameters, (for example, calculation time or correspondence costs) or, to approve presumptions. The refined models can be utilized, to build our trust in the nature of our plan, before execution. We looked at the execution of the parallel program, with its execution demonstrate. Doing this can help both to recognize execution mistakes and to enhance the nature of the model.

An execution, display gives the data around one part of a calculation outline: its normal parallel execution. We can utilize this data, when it is consolidated with assessments of execution cost and so on, to settle on educating decisions, between outline choices. The execution models created in this paper, give a premise to assessing these tradeoffs. Plainly, the decision of the most limited way calculation, for a specific issue will include complex tradeoffs between adaptability, versatility, execution and usage unpredictability.

REFERENCES

1. T. T. Tong-Wook Shinn, "Combining All Pairs Shortest Paths and All Pairs Bottleneck Paths Problems," *optimization and its Application in learning Theory,* pp. 226-236, 2014
2. F. Le Gall, "Faster Algorithms for Rectangular Matrix Multiplication.," in *In: Proc. 53rd FOCS*, 2012
3. R. Wiiliams, "The Polynomial Method and All-Pairs Shortest," *Advanced Complexity Theory,* vol. vol 2, no. 1, pp. 453-568, 2016
4. A. K. Sangaiah1, "An Investigation of Dijkstra and Floyd Algorithms in National City Traffic Advisory Procedures," *International Journal of Computer Science and Mobile Computing,* vol. III, no. 2, pp. pp 124-138, 2014
5. B. D. a. G. F, "A new approach to dynamic all pairs shortest paths," *Journal of the ACM (JACM),* vol. 51, no. 6, pp. 968-992, 2004
6. C. B. Johnson, "Efficient Algorithms for Shortest Paths in Sparse Networks," *Journal of ACM,* vol. 24, no. 24, pp. 1-13, 2016
7. R. Williams, "Faster all-pairs shortest paths via circuit complexity," in *Proceeding STOC '14 -Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 2014
8. N. Alon., "On the Exponent of the All Pairs Shortest Path Problem," *Journal of Computer and System Sciences,* vol. 3, no. 6, pp. 50-62, 2017
9. C. Y. P. Narsingh Deo, "Shortest-path algorithms: Taxonomy and annotation," *Networksw on International journal,* vol. 15, no. 4, pp. 678-690, 2011
10. "Network Theoretic Analysis of Human Brain Network," *FMRI Techniques and Protocols,* vol. 119, pp. 283-314, 2016
11. T. S. L. Vasiliki Kalavri, "The shortest path is not always a straight line: leveraging semi-metricity in network analysis," *Journal Proceedings of the VLDB Endowment,* vol. 9, no. 9, pp. 672-683, 2016.