

Complexity Analysis and Performance Optimization of Distributed Computing Workflows: From Theory to Practice

Yi Gu, Ph.D. candidate, Dept of Computer Science, University of Memphis
Research under Prof. Qishi Wu

I. PROBLEM AND MOTIVATION

The advance of supercomputing technology is expediting the transition in various basic and applied sciences from traditional laboratory-controlled experimental methodologies to modern computational paradigms involving complex numerical model analyses and extreme-scale simulations. These computation-based simulations and analyses have become an essential research and discovery tool in next-generation scientific applications and are producing colossal amounts of simulation data. Other scientific data of similar scales generated in broad science communities include real-world environmental observations such as satellite climate data [1] and multimodal sensor data, and high-throughput experimental measurements such as Spallation Neutron Source [2] and Large Hadron Collider [3].

No matter which type of scientific data is considered, one crucial task would be to develop an effective and efficient end-to-end solution for geographically distributed users to transfer, process, visualize, analyze, and synthesize the data in heterogeneous network environments for collaborative research [4], [5]. The computing tasks of these applications feature complex workflows consisting of many computing modules with intricate inter-module dependencies and require the use of a wide range of expensive resources including supercomputers, PC clusters, high-end workstations, and experimental facilities [6]. Typically, these resources are deployed at various research institutes and national laboratories, and are provided to application users through wide-area network connections that may span through several countries [2], [7], hence inevitably exhibiting an inherent dynamic nature in their accessibility, availability, capacity, and reliability. As new computing and networking technologies rapidly emerge, enabling functionalities are progressing at an ever-increasing pace, unfortunately, so are the dynamics, scale, heterogeneity, and complexity of the networked computing environments. Application users need to manually configure and run computing tasks over networks on their own, oftentimes resulting in unsatisfactory performance in such diverse and dynamic environments. As thus, the science community has identified an urgent need to support such distributed workflows and optimize their end-to-end performance in order to ensure the success of mission-critical e-science and maximize the utilization of massively distributed resources.

We investigate the problem of mapping computing workflows structured as either linear pipelines or Directed Acyclic Graphs (DAGs) to heterogeneous network environments with the goal of automating workflow executions and optimizing their end-to-end performance in terms of Minimum End-to-end Delay (MED) for unitary-input applications (i.e. one-time processing) and Maximum Frame Rate (MFR) for serial-feed applications (i.e. streaming processing). We construct rigorous cost mod-

els for workflows and networks, and formulate the workflow mapping as a set of optimization problems with different objectives and mapping constraints. For each of these problems, we conduct in-depth analysis of computational complexity and develop either an optimal or an efficient heuristic solution [8]–[14]. The performance superiority of the proposed approaches is illustrated by extensive simulation-based comparisons with existing algorithms and further verified by large-scale experiments on real-life scientific workflows through effective system implementation and deployment in real networks [15], [16].

II. BACKGROUND AND RELATED WORK

Many e-science applications face a major performance optimization problem of mapping/scheduling computing workflows with complex execution dependencies in distributed network environments. This problem has attracted much attention from researchers in various disciplines [17]–[32] and continues to be the focus of distributed computing due to its theoretical significance and practical importance.

In the early years, most research efforts were focused on static workflow mapping on multiprocessors that are considered as identical/homogeneous resources [23], [33]. Over the years, workflow mapping problems in heterogeneous environments have been increasingly identified and investigated by many researchers [20], [34], [35]. However, some of these studies only considered independent tasks in the workflow [35] or assumed independent/fully-connected nodes in the network [20], [34], which may not be sufficient to model the complexity of real applications. Recently, a significant number of efforts have been devoted to workflow mapping or task scheduling in grid environments under different mapping and resource constraints. Grid initiatives and projects, such as ASKALON [32], Nimrod/G [36], Globus Toolkit [37], Pegasus [38] and Condor/DAGMan [39], provide toolkits and infrastructures to deploy grid-based scientific computing systems. Similar mapping problems are also studied in the context of stream processing placement [40], sensor networks [41] and bioinformatics [42].

Many workflow mapping or task scheduling problems in grid environments or multiprocessor systems assume independent tasks, complete networks, homogeneous resources, and particular mapping constraints. Our work differs from the aforementioned ones in several aspects: (i) we generalize and formulate the workflow mapping problem based on realistic cost models; (ii) we consider computer networks of arbitrary topology with heterogeneous nodes and links; (iii) we investigate resource sharing dynamics among concurrent module executions and data transfers; (iv) we provide thorough complexity analysis and rigorous stability analysis; (v) we conduct extensive workflow mapping experiments in both simulated and real network environments for model validation and performance valuation.

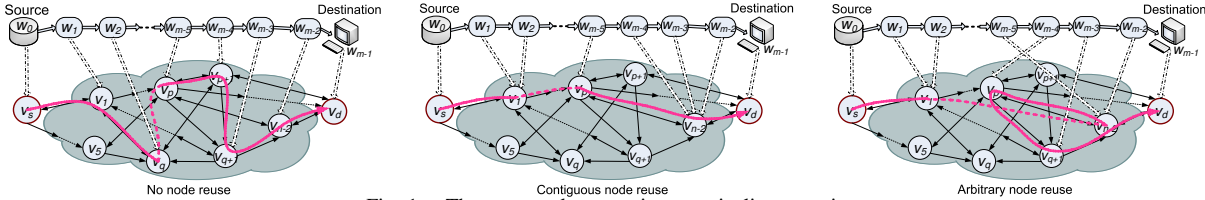


Fig. 1. Three network constraints on pipeline mapping.

III. APPROACHES AND UNIQUENESS

A. Cost Models and Problem Definition

We model the workflow as a Directed Acyclic Graph (DAG) $G_w = (V_w, E_w)$, $|V_w| = m$, where vertices represent computing modules and edges represent execution dependency and data transfer between modules. Module w_j receives a data input from each of its preceding modules and performs a predefined computing routine whose complexity is modeled as a function $\lambda_{w_j}(\cdot)$ on the aggregate input data size z_{w_j} . Once the routine is executed successfully in its entirety, module w_j sends a different data output to each of its succeeding modules.

We model the computer network as a weighted graph $G_c = (V_c, E_c)$ with arbitrary topology, consisting of $|V_c| = n$ nodes interconnected by $|E_c|$ overlay links. We use a normalized variable p_i to represent the overall processing power of node v_i without specifying its detailed system resources. The link $l_{i,j}$ between nodes v_i and v_j has Bandwidth (BW) $b_{i,j}$ and Minimum Link Delay (MLD) $d_{i,j}$. We specify a pair of source and destination nodes (v_s, v_d) to run the start module w_0 and the end module w_{m-1} , respectively, and further assume that w_0 serves as a data source without any computation to supply all initial data needed by the application and w_{m-1} performs a terminal task (e.g. display) without any further data transfer.

We use $T_{\text{exec}}(w, v)$ to denote the execution time of module w on node v :

$$T_{\text{exec}}(w, v) = \sum \frac{\alpha(t) \cdot \delta_w(t)}{p}, \quad \forall w \in V_w, v \in V_c, \quad (1)$$

where $\alpha(t)$ is the number of concurrent modules on node v during Δt , $\delta_w(t) = \frac{p}{\alpha(t)} \Delta t$ denotes the amount of partial module execution completed during time interval $[t, t + \Delta t]$ when $\alpha(t)$ remains unchanged, and $\lambda_w(z_w) = \sum \delta_w(t)$ is the total computational requirement of module w . Similarly, we use $T_{\text{tran}}(e, l)$ to denote the data transfer time of dependency edge e over network link l :

$$T_{\text{tran}}(e, l) = \sum \frac{\beta(t) \cdot \delta_e(t)}{b} + d, \quad \forall e \in E_w, l \in E_c, \quad (2)$$

where $\beta(t)$ is the number of concurrent data transfers over link l during Δt , $\delta_e(t) = \frac{b}{\beta(t)} \Delta t$ denotes the amount of partial data transfer completed during time interval $[t, t + \Delta t]$ when $\beta(t)$ remains unchanged, and $z_e = \sum \delta_e(t)$ is the total data transfer size of dependency edge e . Due to the dynamics in concurrent workload on nodes and concurrent traffic over links, both $\alpha(t)$ and $\beta(t)$ are time-varying in nature and their distributions generally do not exist in a continuous form, which renders standard mathematical solvers inapplicable.

End-to-end Delay (ED) or latency, is the total completion time of the entire workflow. Once a mapping scheme is determined, ED is calculated as the total time cost incurred on the critical path (CP), i.e. the longest path in the DAG. We denote the set of contiguous modules on the CP that are allocated to

the same node as a “group”, and refer to those modules located on the CP as “critical” modules and others as “branch” or “non-critical” modules. A general mapping scheme divides the CP into q ($1 \leq q \leq m$) contiguous groups g_i , $i = 0, 1, \dots, q-1$ of critical modules and maps them to a network path P of not necessarily distinct q nodes, $v_{P[0]}, v_{P[1]}, \dots, v_{P[q-1]}$ from source $v_s = v_{P[0]}$ to destination $v_d = v_{P[q-1]}$. The ED of a mapped workflow is calculated as:

$$\begin{aligned} T_{\text{ED}}(\text{CP mapped to a path } P \text{ of } q \text{ nodes}) &= T_{\text{exec}} + T_{\text{tran}} \\ &= \sum_{i=0}^{q-1} T_{g_i} + \sum_{i=0}^{q-2} T_{e(g_i, g_{i+1})} = \sum_{i=0}^{q-1} \left(\sum_{j \in g_i, j \geq 1} \left(\sum \frac{\alpha_j(t) \cdot \delta_{w_j}(t)}{p_{P[i]}} \right) \right) \\ &+ \sum_{i=0}^{q-2} \left(\sum \frac{\beta_{e(g_i, g_{i+1})}(t) \cdot \delta_{e(g_i, g_{i+1})}(t)}{b_{P[i], P[i+1]}} + d_{P[i], P[i+1]} \right), \end{aligned} \quad (3)$$

where $e(g_i, g_{i+1})$ denotes the dependency edge from group g_i to g_{i+1} mapped to link $l_{P[i], P[i+1]}$ between nodes $v_{P[i]}$ and $v_{P[i+1]}$.

Frame Rate (FR) or throughput, i.e. the inverse of the global Bottleneck Time (BT) of a workflow, is the data production rate at the last module. We maximize FR to produce the smoothest data flow in streaming applications. The BT T_{BT} could be either on a computing module or a dependency edge, computed as:

$$\begin{aligned} T_{\text{BT}}(G_w \text{ mapped to } G_c) &= \max_{\substack{w_i \in V_w, e_{j,k} \in E_w \\ v_{j'} \in V_c, l_{j',k'} \in E_c}} \left(\frac{T_{\text{exec}}(w_i, v_{j'})}{T_{\text{tran}}(e_{j,k}, l_{j',k'})} \right) \\ &= \max_{\substack{w_i \in V_w, e_{j,k} \in E_w \\ v_{j'} \in V_c, l_{j',k'} \in E_c}} \left(\frac{\sum \frac{\alpha_i(t) \cdot \delta_{w_i}(t)}{p_{j'}}}{\sum \frac{\beta_{j,k}(t) \cdot \delta_{e_{j,k}}(t)}{b_{j',k'}} + d_{j',k'}} \right). \end{aligned} \quad (4)$$

We assume that the inter-module communication cost on the same node is negligible. Since the execution start time of a module depends on the availability of its input data, the modules assigned to the same node may not run simultaneously. This is also true for concurrent data transfers over the same link. The mapping problem is formally defined as follows:

Definition 1: Given a DAG-structured computing workflow $G_w = (V_w, E_w)$ and a heterogeneous computer network $G_c = (V_c, E_c)$, we wish to find a mapping scheme that assigns each computing module to a network node such that the mapped workflow achieves:

$$\text{MED} = \min_{\text{all possible mappings}} (T_{\text{ED}}), \quad \text{MFR} = \max_{\text{all possible mappings}} \left(\frac{1}{T_{\text{BT}}} \right).$$

B. Pipeline Mapping [8]–[11]

Linear pipelines are a special case of the general DAG-structured workflows. In pipeline mapping, we categorize the problem into six classes with two optimization objectives and three network constraints as shown in Fig. 1: MED/MFR with No Node Reuse (NNR), Contiguous Node Reuse (CNR), or Arbitrary Node Reuse (ANR). Here, “contiguous node reuse” means that multiple contiguous modules along the pipeline may run on the same node and “arbitrary node reuse” imposes no restriction on node reuse. These mapping problems and their

complexities are tabulated in Fig. 2.

Objective Function	Minimum End-to-end Delay	Maximum Frame Rate
No Node Reuse	NP-complete	NP-complete
Contiguous Node Reuse	NP-complete	NP-complete
Arbitrary Node Reuse	Polynomial (Dyn. Prog.)	NP-complete

Fig. 2. Complexities of six pipeline mapping problems.

We propose a set of mapping algorithms, *Efficient Linear Pipeline Configuration* (ELPC), for pipeline mapping under different constraints, in which a polynomial-time optimal solution is designed to solve MED-ANR using a Dynamic Programming (DP)-based procedure. Let $T^{j-1}(v_i)$ denote the MED with the first j modules mapped to a path from the source node v_s to node v_i in the network. We have the following recursion leading to the final solution $T^{m-1}(v_d)$:

$$T^{j-1}(v_i) = \min_{j=2 \text{ to } m, v_i \in V} \left(T^{j-2}(v_i) + \frac{\lambda_{w_{j-1}}(z_{j-2,j-1})}{p_i} \right), \quad (5)$$

$$\min_{v_u \in \text{pre}(v_i)} \left(T^{j-2}(v_u) + \frac{\lambda_{w_{j-1}}(z_{j-2,j-1})}{p_i} + \frac{z_{j-2,j-1}}{b_{u,i}} + d_{u,i} \right),$$

where $\text{pre}(v_i)$ denotes the set of preceding neighbor nodes of node v_i . Since there is no concurrent node sharing in pipeline mapping with an unitary input, Eq. 5 does not contain α , β , and δ . As shown in Fig. 3, modules are listed in order along the horizontal axis and nodes are arranged along the vertical axis. Each cell $T^{j-1}(v_i)$ in the DP table represents an optimal mapping solution that maps the first j modules in the pipeline to a path between v_s and v_i in the network.

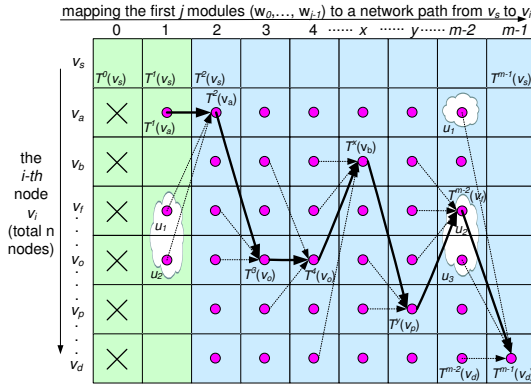


Fig. 3. Construction of 2D table based on DP.

We show that MED/MFR-NNR/CNR are NP-complete by reducing to them from the problem of finding two vertex-disjoint paths in a directed graph and also prove that these problems cannot be approximated by any constant factor, unless $P = NP$ [9], [10]. We prove MFR-ANR to be NP-complete by reducing from the Widest path with Linear Capacity Constraints (WLCC) problem [43]. We develop heuristic solutions by adapting the optimal DP-based method for MED-ANR to the rest five NP-complete problems with appropriate modifications: for MED-CNR, we skip the cell of a neighbor node (slanted incident link) in the left column whose solution (path) involves the current node to ensure a loop-free path; for MED-NNR, we further skip the immediate left neighbor cell (horizontal incident link) to ensure that the current node has never been reused. The steps for MFR problems are similar to those for their corresponding MED problems, but with focus on BT

instead of ED as follows:

$$T_{\text{BT}}^{j-1}(v_i) = \min_{j=2 \text{ to } m, v_i \in V} \left(\max \left(\frac{T_{\text{BT}}^{j-2}(v_i), \alpha_{j-1}(t) \cdot \delta_{w_{j-1}}(t)}{\sum p_i} \right), \min_{v_u \in \text{pre}(v_i)} \left(\max \left(\frac{T_{\text{BT}}^{j-2}(v_u), \alpha_{j-1}(t) \cdot \delta_{w_{j-1}}(t)}{\sum p_i}, \frac{\beta_{j-2,j-1}(t) \cdot \delta_{e_{j-2,j-1}}(t)}{b_{u,i}} + d_{u,i} \right) \right) \right). \quad (6)$$

C. DAG-structured Workflow Mapping [13], [14], [44]

Before proceeding to workflow mapping, we wish to determine whether or not a workflow stabilizes and if it does, how long it takes to stabilize. We make conjectures upon three possibilities of stability: (i) stable, if the last module stabilizes and produces final outputs at a constant rate; (ii) semi-stable, if the interval of final outputs exhibits a periodic pattern; and (iii) unstable, if the output interval is completely unpredictable. We rigorously prove that a one-to-one mapped workflow stabilizes after the global bottleneck time T_{BT} of the entire workflow, and extend this stability result to arbitrarily mapped workflows [14].

Mapping DAG-structured workflows to heterogeneous networks is well known to be NP-complete in a general form [45]. For MED, we first design an algorithm, *extED*, to calculate the exact execution time of each module and data transfer time over each edge by determining the number of concurrent module executions and data transfers, respectively. We then propose an *improved Recursive Critical Path* (imprCP) algorithm [13], which recursively chooses the Critical Path (CP) based on the previous round of calculation and maps it to the network using a DP-based procedure until the results converge to an optimal or suboptimal point as illustrated in Fig. 4. For those branch modules, we develop an A^* search- and Beam Search-based heuristic algorithm to find their corresponding mapping nodes.

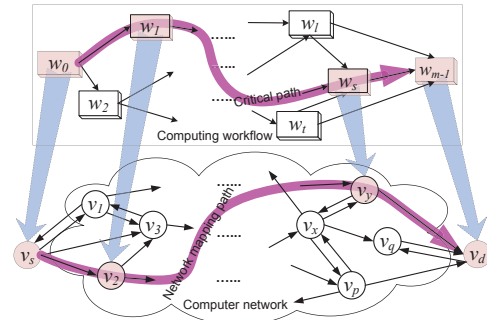


Fig. 4. Map a CP in the workflow to a path in the network.

For MFR, we propose a *Layer-oriented DP* mapping algorithm, LDP, to achieve MFR by identifying and minimizing the global BT T_{BT} of the workflow [14]. The key idea of LDP is to first sort a DAG-structured workflow in a topological order and then map computing modules to network nodes on a layer-by-layer basis, while taking into consideration both module dependency and node connectivity as illustrated in Fig. 5, where the horizontal coordinates represent the sequence numbers of topologically sorted modules (using the priority to break a tie within the same layer), and the vertical coordinates represent the network nodes starting from source to destination. We also propose a greedy version of LDP, referred to as Greedy LDP, to reduce the search complexity of the original LDP algorithm.

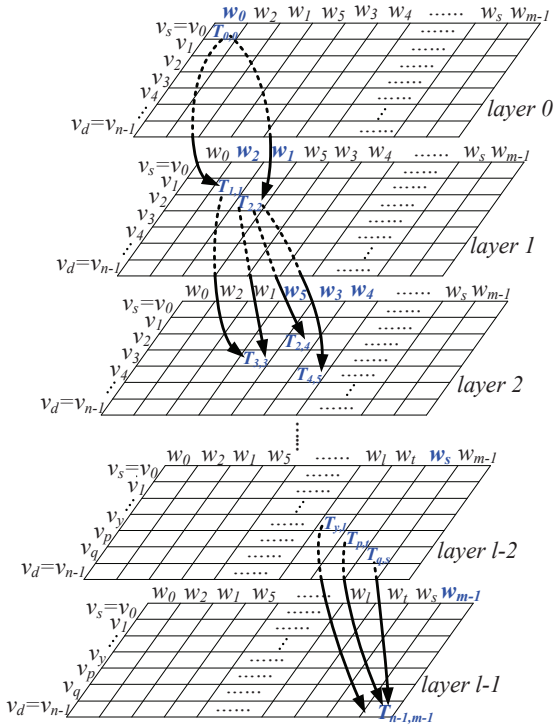


Fig. 5. The DP table with separated layers in LDP.

We also successfully decentralize these mapping algorithms to improve the scalability and adapt them to faulty network environments to provide a guaranteed level of reliability [44] in addition to satisfying the end-to-end optimization goals.

IV. RESULTS AND CONTRIBUTIONS

For pipeline mapping, we conduct simulation-based performance comparison between ELPC, Streamline [27], and Naive Greedy algorithms with various mapping constraints under 20 different problem sizes, indexed from 1 to 20. We plot the performance measurements of MED and MFR produced by these three algorithms in Figs. 6, 8, and 10, and Figs. 7, 9 and 11, respectively. We observe that ELPC exhibits comparable or superior performances in terms of MED and MFR over the other two algorithms in all the cases we studied.

For DAG-structured workflow mapping, we adapt and implement three existing methods for comparison, namely, Greedy A* [41], Streamline [27], and Greedy, and conduct performance evaluations using both simulations and experiments.

In the simulations, we generate workflows and networks by randomly varying the parameters within a suitably selected range of values, represented by a four-tuple $(m, |E_w|, n, |E_c|)$: m modules, $|E_w|$ dependency edges, n nodes, and $|E_c|$ network links. The MED measurements in 15 problem sizes indexed from 1 to 15 are plotted in Fig. 12, where we observe that the proposed impRCP algorithm consistently outperforms all other mapping algorithms and the average MED speedup over the other algorithms is around 10%. The speedup of solution A over solution B is defined as $|\frac{B-A}{B}| \times 100\%$. We also investigate the MFR performance of the proposed Greedy LDP in 20 problem sizes indexed from 1 to 20 and plotted the performance measurements in Fig. 13. Greedy LDP consistently exhibits comparable or superior MFR performances over the other

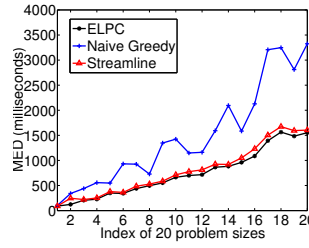


Fig. 6. MED-NNR.

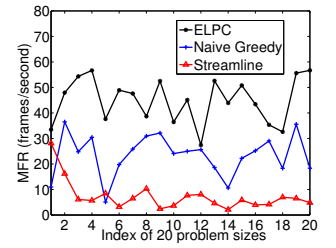


Fig. 7. MFR-NNR.

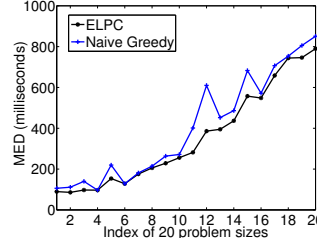


Fig. 8. MED-CNR.

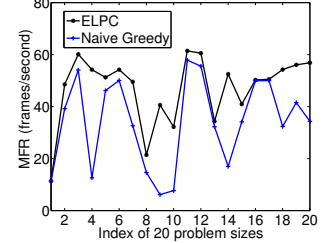


Fig. 9. MFR-CNR.

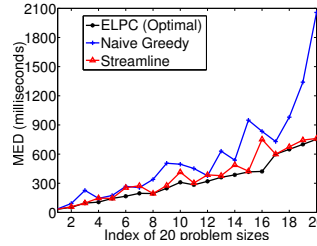


Fig. 10. MED-ANR.

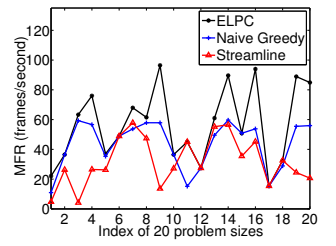


Fig. 11. MFR-ANR.

algorithms, and the average MFR speedup over the other algorithms is around 25%. To further investigate the robustness of Greedy LDP, we randomly generate 500 problem instances with different topologies and sizes. For each algorithm, we calculate and plot the mapping success rate in Fig. 14, defined as $\frac{\text{the number of test cases with valid mapping schemes}}{\text{the total number of test cases}} \times 100\%$. We observe that Greedy LDP has a success rate around 98% while the other algorithms have a success rate around 80%, which indicates that Greedy LDP has a stable mapping performance due to its consideration of module dependency and network connectivity.

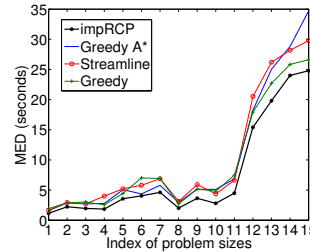


Fig. 12. Simulation-based MED comparison.

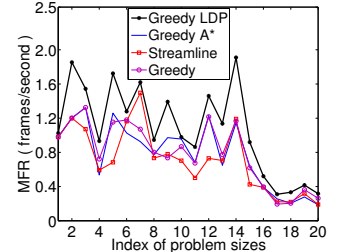


Fig. 13. Simulation-based MFR comparison.

In the experiments, we set up a wide-area network testbed consisting of 12 PC workstations with different hardware configurations. We create an arbitrary network topology by configuring a different firewall setting on each computer and using the Linux traffic control command “tc” to allocate a different bandwidth along each overlay link. We implement and deploy a Condor/DAGMan-based workflow management

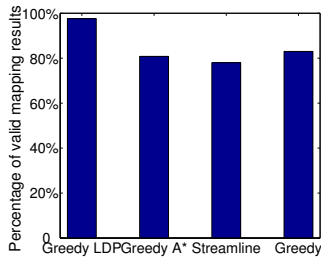


Fig. 14. Success rate in 500 mapping test cases.

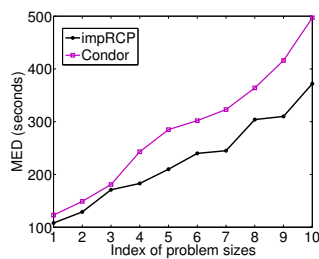


Fig. 15. MED comparison between impRCP and default Condor.

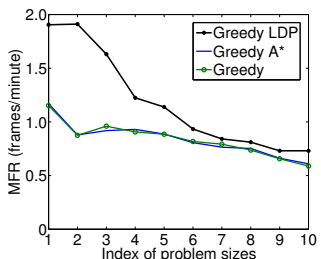


Fig. 16. MFR comparison between Greedy LDP, Greedy A* and Greedy.

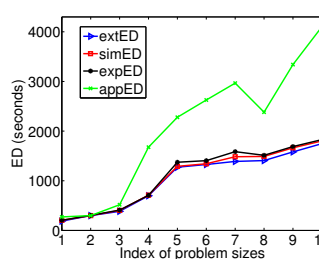


Fig. 17. Comparison among extED, simED, expED and appED results.

system, Scientific Workflow Automation and Management Platform (SWAMP) [15], [16], in the network tested to evaluate the performance of different mapping algorithms.

We integrate the proposed impRCP and Greedy LDP algorithms into SWAMP to run a large-scale scientific application, Spallation Neutron Source (SNS) [2], in a real network environment. The SNS data analysis process includes two basic modules, i.e. data rotation and data rebinning, and six auxiliary modules. The experimental dataset used in this study contains 160 energy slices, each of which contains 61 runs of reduced data. For each energy slice, we create a separate workflow that rotates the data of 61 runs in parallel and then performs a concatenation operation on the rotated data. We run a single component SNS workflow with 10 different input data sizes and measure its MED using impRCP and the Condor/DAGMan's default mapping scheme. Since the default Condor/DAGMan uses a centralized data forwarding mechanism, we adopt Stork in SWAMP to implement a distributed execution model for impRCP. The MED measurements plotted in Fig. 15 show that impRCP consistently achieves a smaller MED than the default Condor/DAGMan. The differences between these two methods are not significant for small input data sizes because Stork itself runs as an individual job and introduces extra overhead during file transfer, but the advantage of impRCP becomes more obvious for large input data sizes because impRCP optimizes the entire workflow mapping and avoids redundant file transfers between execution and submission nodes in Condor/DAGMan. We run SNS streaming experiments on the same datasets to compare MFR among different mapping algorithms. The performance curves produced by Greedy LDP, Greedy A* and Greedy are plotted in Fig. 16. Note that Streamline cannot find a feasible mapping scheme in some cases and hence its performance measurements are not plotted.

To verify the accuracy of extED, we compare it with: (i) appED, an approximate estimate used in [12]; (ii) SDEDS

simulation program in [46], which overlays a workflow on a network based on a given mapping scheme and visually illustrates the dynamic execution process as shown in Fig. 18, where squares, circles, and shadowed rectangles represent modules, nodes, and datasets being processed, respectively; (iii) expED, an experimental measurement using SWAMP.

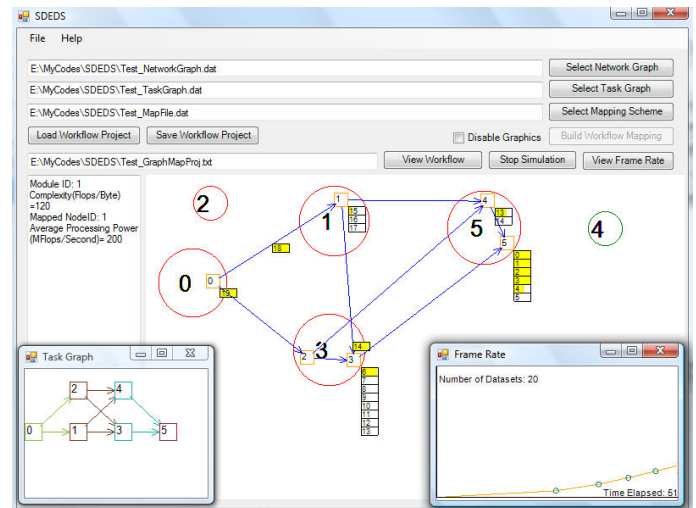


Fig. 18. A running example of a small-scale workflow in SDEDS.

We randomly generate 10 cases, in each of which, we run impRCP to compute a mapping scheme, based on which, we calculate the theoretical values of extED and appED. We then simulate the execution of the mapped workflow in SDEDS to collect simED measurements, and also deploy it through SWAMP in the real network to collect expED measurements. We plot the ED calculations or measurements in Fig. 17 and observe that extED is very close to the SDEDS-based simED and SWAMP-based expED, which indicates (i) the accuracy of the extED calculation, (ii) the validity of the cost models, (iii) the accuracy of the objective functions, and (vi) the correctness of the SDEDS and SWAMP implementation. The appED solution requires much less execution time, and hence could be applied to applications that do not require a high level of accuracy but a prompt response.

In sum, our work lays down a solid mathematical foundation for the analysis and control of system dynamics of large-scale scientific workflows: (i) The complexity analysis (including both NP-completeness and non-approximability proofs) provides a deep insight into the difficulty of workflow mapping problems, which essentially arises from the topological matching nature in the spatial domain, and is further compounded by the resource sharing complicacy in the temporal dimension. (ii) The stability analysis helps us understand the conditions under which a distributed workflow system stabilizes and provides practical guidelines to the design of stable workflow systems. (iii) The mapping optimization solutions represent the state of the arts in this domain and their low polynomial-time computational complexity ensures the scalability of the system in coping with large network sizes and complex workflow structures. (iv) The workflow system with these integrated mapping algorithms has demonstrated salient features in improving the end-to-end performance of real-life scientific applications in wide-area network environments.

REFERENCES

- [1] Climate and Carbon Research Institute. <http://www.ccs.ornl.gov/CCR>.
- [2] Spallation Neutron Source. <http://neutrons.ornl.gov>, <http://www.sns.gov>.
- [3] Relativistic Heavy Ion Collider. <http://www.bnl.gov/rhic>.
- [4] "The office of science data-management challenge," Mar.-May 2004, report from the DOE Office of Science Data-Management Workshops. Technical Report SLAC-R-782, Stanford Linear Accelerator Center.
- [5] NSF Grand Challenges in eScience Workshop, 2001. <http://www2.evl.uic.edu/NSF/index.html>.
- [6] "High-performance networks for high-impact science," Aug. 13-15 2002, report of the High-Performance Network Planning Workshop, <http://DOECollaboratory.pnl.gov/meetings/hpnpw>.
- [7] Earth Simulator Center. <http://www.jamstec.go.jp/esc>.
- [8] Y. Gu, Q. Wu, A. Benoit, and Y. Robert, "Brief announcement: Complexity analysis and algorithm design for optimal pipeline configuration in distributed network environments," in *Proc. of the 28th Annual ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing (PODC)*, Calgary, Canada, Aug. 10-12 2009.
- [9] Q. Wu and Y. Gu, "Optimizing end-to-end performance of data-intensive computing pipelines in heterogeneous network environments," *Journal of Parallel and Distributed Computing*, in press, 2010.
- [10] Y. Gu, Q. Wu, A. Benoit, and Y. Robert, "Optimizing end-to-end performance of distributed applications with linear computing pipelines," in *Proc. of the 15th Int. Conf. on Para. and Dist. Sys.*, Shenzhen, China, Dec. 8-11 2009.
- [11] Y. Gu and Q. Wu, "Optimizing distributed computing workflows in heterogeneous network environments," in *Proc. of the 11th Int. Conf. on Distributed Computing and Networking*, Kolkata, India, Jan. 3-6 2010.
- [12] Q. Wu and Y. Gu, "Supporting distributed application workflows in heterogeneous computing environments," in *Proc. of the 14th IEEE Int. Conf. on Parallel and Distributed Systems*, Melbourne, Australia, Dec. 2008, pp. 3-10.
- [13] Q. Wu, Y. Gu, Y. Liao, X. Lu, Y. Lin, and N. Rao, "Latency modeling and minimization for large-scale scientific workflows in distributed network environments," in the *44th Annual Simulation Symposium (ANSS11), part of the 2011 Spring Simulation Multiconference (SpringSim11)*, Boston, MA, Apr. 4-7 2011.
- [14] Y. Gu and Q. Wu, "Maximizing workflow throughput for streaming applications in distributed environments," in *Proc. of the 19th Int. Conf. on Comp. Comm. and Net.*, Zurich, Switzerland, August 2010.
- [15] Q. Wu, Y. Gu, X. Lu, M. Zhu, P. Brown, W. Lin, and Y. Liu, "On optimization of scientific workflows to support streaming applications in distributed network environments," in *Proc. of the 5th Workshop on Workflows in Support of Large-Scale Science (in conjunction with SC 2010)*, New Orleans, LA, Nov. 14 2010.
- [16] Q. Wu, M. Zhu, X. Lu, P. Brown, Y. Lin, Y. Gu, F. Cao, and M. Reuter, "Automation and management of scientific workflows in distributed network environments," in *Proc. of the 6th Int. Workshop on Sys. Man. Tech., Proc., and Serv.*, Atlanta, GA, April 19 2010.
- [17] R. Bajaj and D. Agrawal, "Improving scheduling of tasks in a heterogeneous environment," *IEEE Trans. on Parallel and Distributed Systems*, vol. 15, pp. 107-118, 2004.
- [18] S. Annie, H. Yu, S. Jin, and K.-C. Lin, "An incremental genetic algorithm approach to multiprocessor scheduling," *IEEE Trans. on Para. and Dist. Sys.*, vol. 15, pp. 824-834, 2004.
- [19] C. Boeres, J. Filho, and V. Rebello, "A cluster-based strategy for scheduling task on heterogeneous processors," in *Proc. of 16th Symp. on Computer Architecture and High Performance Computing*, 2004, pp. 214-221.
- [20] H. Topcuoglu, S. Hariri, and M. Wu, "Performance effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. on Parallel and Distributed Systems*, vol. 13, no. 3, 2002.
- [21] A. Bashir, V. Susarla, and K. Vairavan, "A statistical study of the performance of a task scheduling algorithm," *IEEE Trans. on Computers*, vol. 32, no. 12, pp. 774-777, Dec. 1975.
- [22] V. Chaudhary and J. Aggarwal, "A generalized scheme for mapping parallel algorithms," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 3, pp. 328-346, May 1993.
- [23] A. Gerasoulis and T. Yang, "A comparison of clustering heuristics for scheduling DAGs on multiprocessors," *JPDC*, vol. 16, no. 4, pp. 276-291, Dec. 1992.
- [24] B. Shirazi, M. Wang, and G. Pathak, "Analysis and evaluation of heuristic methods for static scheduling," *J. of Parallel and Distributed Computing*, no. 10, pp. 222-232, 1990.
- [25] L. Chen and G. Agrawal, "Resource allocation in a middleware for streaming data," in *Proc. of the 2nd Workshop on Middleware for Grid Comp.*, Toronto, Canada, Oct. 2004.
- [26] T. Ma and R. Buyya, "Critical-path and priority based algorithms for scheduling workflows with parameter sweep tasks on global grids," in *Proc. of the 17th Int. Symp. on Computer Architecture on HPC*, 2005, pp. 251-258.
- [27] B. Agarwalla, N. Ahmed, D. Hilley, and U. Ramachandran, "Streamline: a scheduling heuristic for streaming application on the grid," in *Proc. of the 13th Multimedia Comp. and Net. Conf.*, San Jose, CA, 2006.
- [28] J. Cao, S. Jarvis, S. Saini, and G. Nudd, "GridFlow: workflow management for grid computing," in *Proc. of the 3rd IEEE/ACM Int. Symp. on Cluster Computing and the Grid*, May 2003, pp. 198-205.
- [29] M. Rahman, S. Venugopal, and R. Buyya, "A dynamic critical path algorithm for scheduling scientific workflow applications on global grids," in *Proc. of the 3rd IEEE Int. Conf. on e-Sci. and Grid Comp.*, 2007, pp. 35-42.
- [30] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *J. of Future Generation Computer Systems*, vol. 25, no. 5, pp. 528-540, May 2009.
- [31] J. Ullman, "NP-complete scheduling problems," *J. of Computer and System Sciences*, vol. 10, no. 3, pp. 384-393, 1975.
- [32] M. Wiczcerek, R. Prodan, and T. Fahringer, "Scheduling of scientific workflows in the ASKALON grid environment," *ACM SIGMOD, Record J.*, vol. 34, no. 3, pp. 56-62, Sep. 2005.
- [33] Y. Kwok and I. Ahmad, "Dynamic critical-path scheduling: An effective technique for allocating task graph to multiprocessors," *IEEE Trans. on Parallel and Distributed Systems*, vol. 7, no. 5, pp. 506-521, May 1996.
- [34] A. Benoit and Y. Robert, "Mapping pipeline skeletons onto heterogeneous platforms," *JPDC*, vol. 68, no. 6, pp. 790-808, 2008.
- [35] T. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen, and R. Freund, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *JPDC*, vol. 61, no. 6, pp. 810-837, June 2001.
- [36] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G: an architecture for a resource management and scheduling system in a global computational grid," in *Proc. of the 4th Int. Conf./Exhibition on the High Performance Computing in the Asia-Pacific Region*, vol. 1, 2000, pp. 283-289.
- [37] I. Foster, "Globus toolkit version 4: software for service-oriented systems," *J. of Comp. Sci. and Tech.*, pp. 513-520, July 2006.
- [38] E. Deelman, G. Singh, M. Su, J. Blythe, A. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, "Pegasus: a framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming Journal*, vol. 13, pp. 219-237, 2005.
- [39] E. Deelman, S. Callaghan, E. Field, H. Francoeur, R. Graves, N. Gupta, V. Gupta, T. Jordan, C. Kesselman, P. Maechling, J. Mehringer, G. Mehta, D. Okaya, K. Vahi, and L. Zhao, "Managing large-scale workflow execution from resource provisioning to provenance tracking: the cybershake example," in *Proc. of the e-Science Conf.*, Amsterdam, Netherlands, Dec. 2006.
- [40] J. Wolf, N. Bansal, K. Hildrum, S. Parekh, D. Rajan, R. Wagle, and K. Wu, "Job admission and resource allocation in distributed streaming systems," in *Proc. of the 14th Int. Workshop on Job Scheduling Strategies for Parallel Processing*, Rome, Italy, 2009, pp. 169-189.
- [41] A. Sekhar, B. Manoj, and C. Murthy, "A state-space search approach for optimizing reliability and cost of execution in distributed sensor networks," in *Proc. of Int. Workshop on Dist. Comp.*, 2005, pp. 63-74.
- [42] T. Oinn, M. Addis, J. Ferris, D. Marvin, T. Carver, M. Pocock, and A. Wipat, "Taverna: A tool for the composition and enactment of bioinformatics workflows," in *Bioinformatics*, vol. 20, no. 17, 2004, pp. 3045-3054.
- [43] Y. Zhu and B. Li, "Overlay network with linear capacity constraints," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, pp. 159-173, Feb. 2008.
- [44] Q. Wu and Y. Gu, "A distributed workflow mapping algorithm for minimum end-to-end delay under fault-tolerance constraint," in *Proc. of the 16th Int. Conf. on Para. and Dist. Sys.*, Shanghai, China, Dec. 8-10 2010.
- [45] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*. San Francisco: W.H. Freeman and Company, 1979.
- [46] Q. Wu, Y. Gu, and Z. Wang, "Simulation-based analysis of performance dynamics of distributed applications in heterogeneous network environments," in *SpringSim'09: Proc. of the 2009 Spring Simulation Multiconference*, San Diego, CA, March 22-27 2009, pp. 1-8.