Measuring the Effectiveness of Throttled Data Transfers on Data-Intensive Workflows

Ricardo J. Rodríguez[1], Rafael Tolosana-Calasanz[1], and Omer F. Rana[2]

[1] Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, María de Luna 1, 50018 - Zaragoza, Spain
{rjrodriguez, rafaelt}@unizar.es
[2] School of Computer Science & Informatics
Cardiff University - Cardiff, United Kingdom
o.f.rana@cs.cardiff.ac.uk

Abstract. In data intensive workflows, which often involve files, transfer between tasks is typically accomplished as fast as the network links allow, and once transferred, the files are buffered/stored at their destination. Where a task requires multiple files to execute (from different previous tasks), it must remain idle until all files are available. Hence, network bandwidth and buffer/storage within a workflow are often not used effectively. In this paper, we are quantitatively measuring the impact that applying an intelligent data movement policy can have on buffer/storage in comparison with existing approaches. Our main objective is to propose a metric that considers a workflow structure expressed as a Directed Acyclic Graph (DAG), and performance information collected from historical past executions of the considered workflow. This metric is intended for use at the design-stage, to compare various DAG structures and evaluate their *potential* for optimisation (of network bandwidth and buffer use).

1 Introduction

Scientists can make use of workflow techniques, using often a combination of control/data flow graphs, to specify their experiments. There has been an increased interest in data-intensive scientific workflows recently, where the data between tasks in the workflow needs to be migrated across a distributed environment. Often the staging of data between tasks in such workflows is either assumed or the data transfer time is considered to be negligible (compared to task execution). Where data consists of files, previous approaches exploit data location and link bandwidth information to minimise data movement, but the transfer typically involves moving the output data of a task to its successor node as fast as possible after task completion [1]. However, when a task needs multiple files to be made available before it can begin execution, the task remains idle until all the required data files from other predecessor nodes are delivered. It is therefore not how fast each file can be moved to the task, but the interval from the delivery of the first file to the last one that is most significant. Even

if the first file is delivered quickly, the task must still remain idle until others are also available, thereby leading to an ineffective use of *network bandwidth* and *buffer/storage* at the receiving task. If one file arrives too early taking up all of the network bandwidth for one task, it may be at a determent to other tasks (which may have to wait for their data to be delivered) – even though the receiving task still has to wait for other files. Similarly, if there is limited buffer capacity at the receiving task and the buffer needs to be shared between tasks, a quick delivery of one file (while waiting for other files to be delivered) for a task excludes other tasks from using the same buffer.

Therefore, current practice of moving data as early as possible is either: (i) unnecessary when viewed in isolation; or (ii) harmful when viewed in-the-large: due to finite capacity and competing transfers on the same link and buffer. Park & Humphrey [1] proposed a data throttle strategy so that when a task needs multiple input data sets, all of them arrive simultaneously. However, they do not discuss a mechanism for analysing and automatically deriving the throttling rates.

The benefit of using a data throttling strategy depends on the workflow structure, the intermediate data [3] size, and the execution environment (i.e. networking and computational resource characteristics, etc.). Such a strategy is applicable when a workflow task has multiple data inputs (we refer to these as "synchronisation points" in the workflow), enabling some input links to be throttled. We argue that the greater the number of such synchronisation points, the greater the benefit we are likely to see. We quantitatively measure this benefit, capturing it as a metric value that can be used to compare different workflow graph structures (achieving the same outcome). Our metric computation process converts the workflow DAG specification into a Petri net and combines this with intermediate data, computational, and network resource information. Subsequent analysis on the Petri net involves calculation of a "slack" (identifying which input data arrives first (the highest slack) and which arrives last (the slowest slack)). With this slack information, we calculate the value of the metric for each task and then aggregate this for the entire workflow. This paper is structured as follows: Section 2 describes related work in using Petri nets and automated management of data transfers in workflows. Section 3 describes how Petri nets can be used to derive performance analysis from workflow DAGs, leading to the description and calculation of our optimisation metric in Sections 4 and 5 – the main contribution of this paper. Section 6 includes a validation of the approach using the Montage workflow – leading to general conclusions that can be drawn outlined in Section 7.

2 Related Work

Petri nets and their extensions have been widely used for the specification, analysis and implementation of workflows [2]. Van der Aalst and Van Hee [3] de-

[3] Datasets obtained as a consequence of the execution of intermediate steps of a workflow are typically known as intermediate data.

scribe WF-nets for modelling workflows with control operations such as sequence, choice, synchroniser, fork or merge. The types of structural analysis that can be undertaken includes correctness, deadlock analysis or liveness. In our approach a workflow graph (DAG) is mapped to a Petri net representation for subsequent analysis, as described in Section 3.

Data movement has also been a matter of study in a number of parallel and distributed research domains, an optimisation of MPI communications that both i) exploits spatial locality of files, for reducing the communications, and ii) dynamically chooses the best compression method for MPI messages is proposed in [4]. Yu and Buyya [5] classify automated data transfer strategies in workflows as centralised, mediated, and peer-to-peer. A centralised approach is often used when the time for data transfers is much smaller than computations. Taverna [6] typically utilises a centralised data transfer, due to the characteristics of the problems it tackles. In a mediated strategy the locations of intermediate data are managed by a distributed data management system. In a Peer-to-Peer (P2P) data is transferred directly between nodes, reducing both transmission time and the bottleneck caused by other approaches. We focus on data-intensive workflows using a P2P data sharing strategy. The Pegasus workflow system [7] supports both mediated and P2P transfers. In the mediated approach, Pegasus utilises a data replica catalogue that stores the intermediate data generated, so data can be subsequently retrieved rather than recomputed again. However, none of these approaches makes an effective usage of both network bandwidth and buffer space, often focusing instead on attempting to transfer the data as quickly as possible between workflow tasks. In [8], we proposed a method for deriving sub-optimal throttling values in data-intensive workflow DAGs.

3 Background: Petri nets

A *Petri net* [9] (PN) is a mathematical formalism used for the modelling of concurrent and distributed systems. A PN is a bipartite graph, which consists of two set of nodes (places P, transitions T) and directed arcs. An arc connects a place with a transition, or vice versa, but it never connects a place with a place or a transition with a transition. The *initial marking* of place p, denoted as $\mathbf{m_0}(p)$, is the natural number of tokens that a place p may have. A *marking* $\mathbf{m}$ defines a distribution of tokens over the places. Normally, PNs have a graphical notation where places are represented by circles, transitions by bars, tokens by black dots and directed arcs by arrows. A transition of a PN is said to be *enabled* when all its input places (i.e., there exists an arc connecting a place with such a transition) have *tokens*. Once enabled, it can be *fired*. When fired, all these tokens are consumed (i.e., removed from input places) and placed in the output places. The firing of transitions of a PN is non-deterministic, i.e., when multiple transitions are enabled at the same time any of them can be fired.

The PN formalism can be extended by associating transitions with a delay. The delay is characterised by a random variable and represents the elapsed time for firing the transition. Such an extension is called *Stochastic Petri nets*

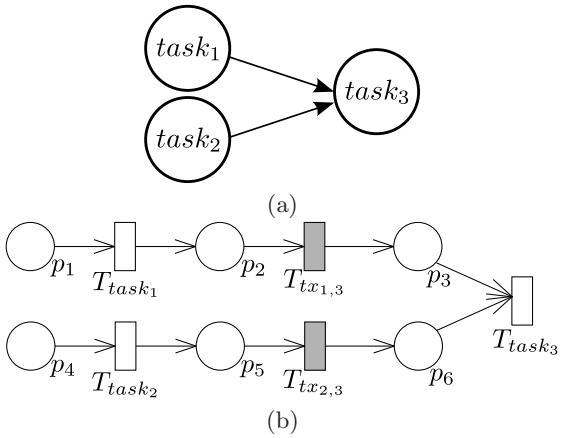

Fig. 1. (a) Workflow tasks and (b) its transformation to PN.

(SPNs) [10], where the random variables follow an exponential distribution. *Stochastic Marked Graphs* (SMGs) are a subset of SPNs, i.e., a SPN with a Marked Graph structure [9]: each place of the SPN has exactly one input and exactly one output arc.

In this paper, we make use of SMGs with exponential random distributions associated with transitions in order to model scientific workflows. In particular, we are interested in workflows expressed as a DAX, which is an XML representation of a DAG that most scientific workflow systems utilise. We automatically convert a DAX into a Petri net representation, in which vertices represent tasks and edges represent data dependencies between them. Fig. 1 depicts how we derive a PN model from a DAG. Fig. 1(a) shows three workflow tasks of a DAG, $task_1$, $task_2$ and $task_3$ and a data-link dependence from $task_1$ to $task_3$ and from $task_2$ to $task_3$, while Fig. 1(b) illustrates the transformation to a PN. Such a PN model fulfills the definition of a SMG model: there are timed transitions and each place has exactly one input and exactly one output arc.

Hence, each task of the workflow DAG is transformed to a place and a transition (represented by a white rectangle), joined by an arc. A task transmission is also transformed to a place and a transition (grey rectangle). For instance, $p_1 \rightarrow T_{task_1}$ represents $task_1$ of the workflow. Note that place p_1 models the input buffer of $task_1$. Finally, the data dependency between $task_1$ and $task_3$ is modelled by adding a place and a transition, p_2 and $T_{tx_{1,3}}$, respectively. Transition $T_{tx_{1,3}}$ represents the time spent in sending output data from $task_1$ to the input buffer of $task_3$ (place p_3). Note that transition T_{task_3}, which represents $task_3$, is not enabled until both places p_3 and p_6 have some token. Such places represent the synchronisation of $task_3$, which is unable to start its execution until it has received output data from $task_1$ and $task_2$.

4 Metric Definition

A task in a workflow DAG cannot start its execution until all its inputs are available. The strategy of receiving these input values as fast as possible is often not appropriate. As some inputs may arrive earlier than others, these inputs have to be buffered locally at the task, resulting in unnecessary use of buffer space. If such buffer space is a shared resource and of limited capacity, it remains blocked by the task, waiting for the remaining data to arrive. Hence, the greater the variation between arrival times of the different input data sets, the greater the inefficiency in buffer use. Intuitively, the objective of an effective data transfer is that each task with multiple inputs has all its data sets arrive simultaneously.

Our approach is therefore relevant for a workflow which has: (i) multiple synchronisation points (identified as tasks in the workflow containing more than one input, where all inputs are needed before the task can begin execution); (ii) difference in arrival times between the different inputs to such synchronisation point. The higher the value of (i) and (ii), the greater the possible optimisation we are likely to see with our approach. Both of these aspects depend on the structure of the workflow and the environment within which a workflow is enacted. Our approach could be used to re-write a workflow DAG that has a *structural imbalance*, i.e. a DAG containing multiple paths whose execution times differ significantly. Such an imbalance [1] may also arise due to a scheduler binding tasks to resources, faults or unexpected performance degradation, a slow network connection, or limited storage for the dataset.

In order to compute the metric, we convert the workflow DAG specification into a Petri net (as explained in Section 3), we subsequently feed the Petri net model with performance information on computational tasks, and network, as well as data size [8]. Petri net theory is subsequently used to analyse the Petri net model, and to obtain *slack* (μ) [11] values (a key concept in our analysis). Intuitively, a *slack* is a positive value associated with each input link to a synchronisation point and captures the time taken for an input data to be delivered to the sync. point. The higher the slack, the greater the probability that input delay will be higher, thereby delaying the execution of the task at the sync. point. A more formal description of a synchronisation point and the associated slack is as follows. Let workflow W be composed of a set of n tasks $T = \{t_1, \ldots, t_n\}$ where each task has m possible inputs, and where W is represented as a cyclic Petri net. A sync. point is defined as a task t_i with multiple inputs, i.e., $T' \subset T$ and $t_i \in T', m > 1$ – with the total number of sync. points being s, i.e., $|T'| = s, s \leq n$.

Let $d_{i,j}, j \in \{1 \ldots m\}$ represent the time taken for input j to arrive at task t_i. As each input arrives at different times, we can determine the value of $max(d_{i,j})$ for a sync. point t_i (i.e. the time taken for the slowest arriving input). Considering the entire workflow W, we can find the slowest path from the input to the output of the workflow, which also represents the workflow makespan – represented as $\sum_{i=1}^{n} max(d_{i,j}) + execution\ time(t_i), \forall_j$. The slack $\mu_{i,j} > 0, j \in \{1 \ldots m\}$, for input j of task t_i, can be calculated as:

$$\mu_{i,j} = \frac{(max_{j=1}^{m}(d_{i,j}) - d_{i,j})}{\sum_{i=1}^{n} max(d_{i,j}) + execution\ time(t_i), \forall_j} \tag{1}$$

The slack, therefore, is a measure of the time arrival of inputs with respect to the critical path of a workflow. Recall that the slack value for the slowest input $max(d_{i,j})$ is always 0. Our metric makes use of *slack* theory and incorporates both structural and execution environment aspects. The metric α can be expressed as:

$$\alpha = s \cdot \sum_{t_i \in T', j \in \{1...m\}} \mu_{i,j} \tag{2}$$

where s represents the number of synchronisation points in the workflow and $\sum_{t_i \in T', j \in \{1...m\}} \mu_{i,j}$, represents the summation of all the slacks that appear in the workflow. The value of α quantifies the potential benefit that a data-throttling strategy may achieve in comparison with a transmit as-fast-as-possible strategy. The metric indicates that the greater the number of tasks with multiple inputs, the greater the potential to save buffer/storage space. The second aspect the metric considers is the summation of all the slack values that appear in the workflow in case a transmit as-fast-as-possible strategy is applied. Ideally, an intelligent transfer strategy would make all the slack values equal to 0.

5 Metric Calculation

Our analysis algorithm and metric calculation is based on Little's law, $L = \lambda \cdot \chi$, which states that the average number of elements in a queue is given by the product of the system's throughput (λ), and the average time spent by the element in the queue (χ). In terms of a Petri net representation, each pair (input place, transition), which represents a pair (input, task) in a workflow, can be seen as a simple queueing system for which Little's law can be directly applied: $\mathbf{m}(p) = \Theta \cdot \delta_t + \mu(p)$, where $\mathbf{m}(p)$ denotes the average number of tokens in place p, Θ represents the system's throughput, δ_t is the average delay associated for the computation of transition t (it models a task), and $\mu(p)$ is the slack value. It should be noted that Little's average queueing time is the addition of the average waiting time due to a possible synchronisation and the average service time, which in this case is δ_t.

In consequence, the first step in the analysis algorithm is to compute the system's throughput Θ, the inverse of the time delay of the critical path. The time delay for the critical path is obtained by solving a Linear Programming Problem (LPP) [12]. The second step is to obtain a special marking (a PN state) of the SMG, which is called *tight marking*, denoted as $\tilde{\mathbf{m}}$. This state fulfils a number of properties, namely i) the tight marking is reachable from the initial

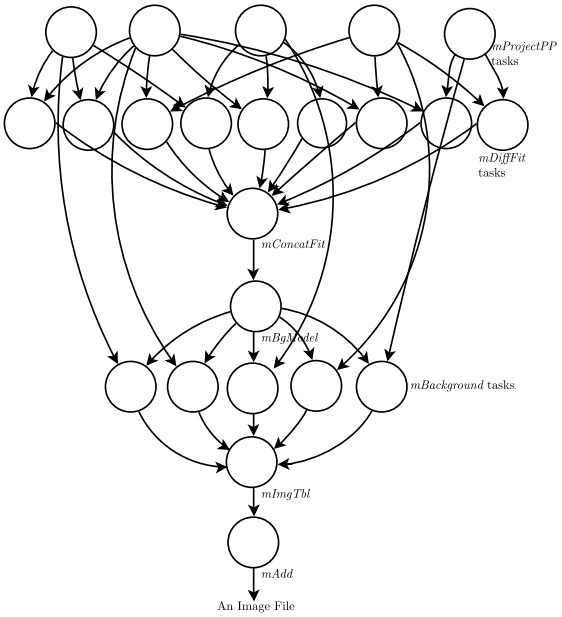

Fig. 2. Montage workflow graph with 25 tasks

marking (initial state), ii) for each input place p, $\tilde{\mathbf{m}}(p) \geq \Theta \cdot \delta$, which is related to Little's Law, and iii) for each transition t, there is at least one input that has no slack, $\tilde{\mathbf{m}}(p) = \Theta \cdot \delta_t$. The tight marking is also computed by a LPP. Finally, for all place p, the slack is easily computed by conducting this algebraic operation, $\mu(p) = \tilde{\mathbf{m}}(p) - \delta_t \cdot \Theta$.

6 Evaluation

In order to evaluate how our metric α measures the effectiveness of utilising a data throttling strategy, we conducted several experiments using the Montage [13] workflow. This workflow is used for creating image mosaics in astrophysics (using data from different scientific instruments) and a specification of Montage structure can be found in Fig. 2. The structure of the workflow is quite regular, therefore the workflow imbalance, if any, must be due to the execution environment. However, Montage workflow structure depends on the number of input files to assemble. We considered 3 versions of Montage, with 25, 50 and 100 tasks, and having each workflow 5, 10 and 100 input files, respectively. A DAX description of this workflow, along with performance information collected from past executions is available at the Pegasus workflow system [7] Web site [4].

We have simulated the enactment of these 3 workflows by making use of the SimGrid [14] tool. We assumed an environment with the number of machines

[4] `https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator`

being the same as the number of nodes in the workflow. As for the network topology, we assumed a single output data-link per host, a network bandwidth of 100Mbps and a latency of 10^{-4}s.

Montage task	**25 tasks**		**50 tasks**		**100 tasks**	
	Mean	σ	**Mean**	σ	**Mean**	σ
mDiffFit	1.7726s	1.1800s	1.5739s	1.2044s	1.9819s	1.6399s
mConcatFit	15.9589s	–	57.5424s	–	226.1896s	–
mBackground	15.0307s	1.1194s	19.0069s	1.4327s	25.1063s	1.8597s
mImgTbl	2.9567s	–	1.4934s	–	4.0822s	–

Table 1. Mean & standard deviation values of buffer waiting time for Montage workflow for $25, 50, 100$ tasks.

For each Montage workflow, Table 1 shows the mean values and standard deviation, σ, of the buffer waiting time of the Montage tasks with multiple input data, namely *mDiffFit*, *mConcatFit*, *mBackground*, and *mImgTbl*. A buffer waiting time was computed as the elapsed time between the arrival of the last and the first input. The results show that for some tasks, the higher the number of synchronisation points, the longer the buffer wait times. The main reason for this is that the synchronisation points of larger Montage workflows have a larger number of inputs in comparison with smaller sized workflows. In particular, this fact can be observed in task *mConcatFit*.

No. tasks	**No. tasks with inputs**	$\sum \mu$	**Metric** α
25	15	1.5173	22.7591
50	32	2.7681	88.5785
100	75	5.3371	400.2842

Table 2. Metric values computed for the considered Montage workflows.

The metric values for each considered Montage workflow are shown in Table 2. The first column indicates the number of total tasks in the workflow, while the second shows the number of tasks with multiple input dependencies. Finally, the third column, $\sum \mu$, represents the summation of slacks (computed as indicated in Section 5 with the MATLAB LP toolkit) and the last column presents the value of the α metric computed. As it can be expected, and it is remarked by the buffer waiting time given in Table 1, the more value of α, the better expected reduction of waiting time by using a data-throttling strategy. For the Montage workflow, it can be then concluded that the higher the number of task in the Montage version, the more important a data throttling strategy will be, so that buffer occupancy can be utilised more effectively.

7 Conclusions

The enactment of data intensive workflows often involves the transfer of files between tasks. Where a task requires multiple files to execute, it must remain idle until all files are available. Hence, an effective and intelligent data transfer strategy ideally would consist in throttling data, so that all input files for a given task arrive simultaneously. In this paper, we study the potential impact that applying such an strategy can have on a workflow, expressed as a Directed Acyclic Graph (DAG). We propose a *quantitative* metric based on the workflow structure, and on performance information derived from past historical executions. We convert the DAG specification into a Petri net model, feed it with such a performance information, and conduct analysis over it to obtain task inputs with *slack*. A slack is a positive value that when computed, appears at a task input that is likely to be idle, waiting for other datasets to arrive. The higher the slack value, the more likely the associated input will have to wait for a longer period. Our metric is proportional to the number of synchronisation points in the workflow and to the sum of all slack values appearing in the workflow. This metric is intended for use at the design-stage, to compare various DAG structures and evaluate their *potential* for optimisation (of network bandwidth and buffer use).

We conducted experiments over 3 Montage workflows, with different sizes. For each synchronisation point in the workflow, we measured the buffer occupancy time and obtained our metric. We observed that the higher the value of the metric, the higher the buffer occupancy time within the Montage workflows. The throttling of data becomes more important with the growing size of the Montage workflow. Due to its characteristics, a Montage workflow with a larger number of tasks suffers from larger buffer occupancy, and this will require more buffering space for the inputs. Although demonstrated through a single workflow our approach is quite general in scope and can be applied to any workflow described using the DAX representation. As future work, we are considering the design and implementation of a tool for workflow design that assists users in their workflow configurations, and helps them determine: i) comparison of different workflows / workflow configurations for potential optimisation, ii) sub-optimal throttling values, iii) performance speed-up or degradation when applying data throttling, and iv) buffer storage needs.

Acknowledgements

This work was supported by the Projects EU FP7 DISC (INFSO-ICT-224498), Spanish CICYT DPI2010-20413, and TIN2010-17905, and by Fundación Aragón I+D. Mr Rodríguez's work was supported by grants from DGA (CONAID) and CAI, "Programa Europa XXI" ref. n. IT 5/11, and Ministerio de Educación, Cultura y Deporte, ref n. TME2011-00255.

References

1. Park, S.M., Humphrey, M.: Data Throttling for Data-Intensive Workflows. In: IEEE International Symposium on Parallel and Distributed Processing. (April 2008) 1–11
2. van der Aalst, W., van Hee, K.: Workflow Management: Models, Methods, and Systems. Volume 1 of MIT Press Books. The MIT Press (2004)
3. Aalst, W.M.P.v.d., Hirnschall, A., Verbeek, H.M.W.E.: An Alternative Way to Analyze Workflow Graphs. In: Proceedings of CAiSE, London, UK, UK, Springer-Verlag (2002) 535–552
4. Filgueira, R., Carretero, J., Singh, D.E., Calderón, A., Nuñez, A.: Dynamic-compi: dynamic optimization techniques for mpi parallel applications. The Journal of Supercomputing **59**(1) (2012) 361–391
5. Yu, J., Buyya, R.: A Taxonomy of Workflow Management Systems for Grid Computing. CoRR **34**(3) (September 2005) 44–49
6. Oinn, T., Greenwood, M., Addis, M., Alpdemir, M.N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M.R., Senger, M., Stevens, R., Wipat, A., Wroe, C.: Taverna: lessons in creating a workflow environment for the life sciences: Research Articles. Concurr. Comput. : Pract. Exper. **18**(10) (2006) 1067–1100
7. Deelman, E., Mehta, G., Singh, G., Su, M., Vahi, K.: Pegasus: Mapping Large-Scale Workflows to Distributed Resources. In: Workflows for eScience. Springer (2007) 376–394
8. Rodríguez, R.J., Tolosana-Calasanz, R., Rana, O.F.: Automating Data-Throttling Analysis for Data-Intensive Workflows. In: Proceedings of CCGrid. (2012) Accepted for publication.
9. Murata, T.: Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE. Volume 77. (April 1989) 541–580
10. Molloy, M.: Performance Analysis Using Stochastic Petri Nets. IEEE Transactions on Computers **C-31**(9) (sept. 1982) 913–917
11. Rodríguez, R.J., Júlvez, J.: Accurate Performance Estimation for Stochastic Marked Graphs by Bottleneck Regrowing. In: Proceedings of the 7th European Performance Engineering Workshop (EPEW). Volume 6342 of LNCS., Springer (September 2010) 175–190
12. Campos, J., Silva, M.: Embedded Product-Form Queueing Networks and the Improvement of Performance Bounds for Petri Net Systems. Performance Evaluation **18**(1) (July 1993) 3–19
13. Berriman, G.B., Deelman, E., Good, J., Jacob, J.C., Katz, D.S., Laity, A.C., Prince, T.A., Singh, G., Su, M.H.: Generating Complex Astronomy Workflows. In Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M., eds.: Workflows for e-Science. Springer London (2007) 19–38
14. Casanova, H., Legrand, A., Quinson, M.: SimGrid: a Generic Framework for Large-Scale Distributed Experiments. In: 10th IEEE International Conference on Computer Modeling and Simulation. (March 2008)