UDK 004.75

B. Bulakh
National Technical University of Ukraine “KPI”

ANALYSIS AND IMPLEMENTATION OF COMPUTING WORKFLOWS FOR SERVICE-ORIENTED CAE/CAD SOFTWARE

This paper is devoted to the architectural aspects of integration of workflow management into service-oriented CAE/CAD software capable to run computations on grid resources. The analysis of possible solutions is based on the workflow execution time estimation. The multi-level workflow concept is also described.

Key words: workflow, SOA, grid computing, simulation.

Розглянуто архітектурні аспекти інтеграції управління робочими процесами в орієнтованій на сервіс CAE/CAD, здатній виконувати обчислення на мережевих ресурсах. Аналіз можливих рішень відбувається на основі оцінки часу виконання робочих процесів. Описано також багаторівневу концепцію робочого процесу.

Ключові слова: робочий процес, орієнтована на сервіс архітектура, мережеві обчислення, моделювання.

Introduction

Development of novel architectural solutions for CAE/CAD systems which can utilize distributed and grid computing and support workflow computations is an urgent problem as well as an analysis of the efficiency of such solutions. Grid computing technology can help to overcome resource limitations for compute-intensive tasks while workflow computation model enable automated execution of custom user computing scenarios instead of fixed computing sequence hard-coded by developers. In order to select optimal architectural decisions for CAD tools supporting workflows a set of metrics for workflow solutions is needed to be provided.

Requirements for application architecture

Competitive CAE/CAD applications should maximally leverage from current IT achievements. The application architecture of any software system is defined by a set of factors, among them are: capabilities of the target hardware/software platform, applied software development technologies, specific work style of end users etc. Wide spreading of multi-core CPUs and high-speed networks, as well as construction of dedicated computing clusters make computing hardware more powerful and decrease networking delays but even powerful single resource may not be sufficient to solve compute-intensive problems. Also the diversity of incompatible OS makes it harder to deploy the software on different platforms. Distributed application architectures and remote computing can help to solve the resource deficit problem as well as provide shared access to unique resources. Cross-platform and standardized solutions play an important role in development of complex systems composed from independent components. Web technologies greatly contribute in spreading of collective work and knowledge exchange support solutions, virtual workplace concept etc. which are common features of many applications now.

The analysis of these trends allows to generalize the requirements for modern CAE/CAD software, among them are: 1) ability to overcome resource limitations during operation with object models of high complexity by migrating the calculations to free remote resources, 2) simplified integration with third-party software inside of software complexes for interdisciplinary research with the help of standardized cross-platform protocols and interfaces, 3) network access to the virtual workplace supporting collective work of engineers.

To satisfy the last requirement a client-server model could be applied. In particular case the interface part can be developed in form of web application accessible from any modern web browser. At the same time to satisfy the other requirements the specific architecture model should be chosen for computing subsystems. It is proposed in [1] to apply the service-oriented architecture (SOA) to satisfy these requirements and the possible ways of integration of grid computing into this architecture were also covered. SOA [2] as the possible solution has the additional advantage: it supports initially the dynamic system configuring based on orchestration of the available services. This makes possible to apply service orchestration for execution of complex computational workflows that can be edited directly by users.

Workflows for computer-aided design

As the fragments of the design process can be considered as workflows the architecture of CAE/CAD tools should contain workflow management system (WfMS [3]) capable to invoke web services in case of service-oriented software design (e.g. WS-BPEL engine).

The workflow $w=(A_w,T_w,D_w)$ here is considered as the union of:

1) the set $A_w \subseteq A$ of computing operations a_i, $i=1..N$, from the certain superset of available operations A ($\forall a_i \in A_w, a_i \in A$);

2) the set of transitions $T_w \subseteq T$ from some superset of allowed transitions T, forming the order of execution of workflow operations ($T_w=(a_s,a_d)$ where $a_s \in A_w$ is the predecessor operation, $a_d \in A_w$ is the successor operation which means that the operation a_d can only be executed after a_s);

3) the set of data channels $D_w \subseteq D$ from some superset of valid (compatible with respect to input and ouput formats) data transfer channels D defining the sequence of data transfers in the workflow ($D_w=(a_s,a_d)$ where $a_s \in A_w$ is the data producer operation, $a_d \in A_w$ is the data consumer operation which means that the output data of a_s operation id fed directly to a_d).

The workflow belongs to the set of valid wokflow configurations $w \in W$ defining the specific workflow model of the WfMS. Sometimes it is more convenient to consider a workflow only from execution order aspect $w_T=(A_w,T_w)$ (called control flow) or only from the data transfer aspect $w_D=(A_w,D_w)$ (called data flow).

The main idea of service-oriented workflows is that web service operations are representing $\{a_i\}$, and WfMS is responsible for their invocations in the correct order specified by T_w as well as for their supply with data according to D_w. This approach is already applied for distributed computing for different applications but its application for CAD tasks still remains not studied well.

Workflow solutions evaluation

The analysis of concrete solutions related to the WfMS integration to the CAD system architecture should be based on some defined criteria. It is possible to mention many metrics that can be applied for comparative analysis of distributed software (internal cohesion, external coupling) as well as workflows and WfMS (cyclomatic complexity, decomposition level, granularity, operation composability etc.). But it should be noted that optimal criteria of such solutions substantially differs at the system developer's point of view (interested in simplicity of software development and deployment through modularity, distributivity and reuse) and at the end user's point of view (interested in simpler interaction with software, extension of its functional capabilities, and high performance). In this paper the quantitative time metrics was chosen as the basis of workflow analysis.

Workflow execution time estimation for workflows of arbitrary configuration is possible, for example, with the help of simulation techniques. But it is also possible to obtain some analytic estimates for the primitive workflow configurations like sequence w_{seq} and parallel flow w_{par} to compare different variants of workflow implementation:

$$t(w_{seq}(a_1,a_2,\ldots,a_N))=\mathop{}\limits_{i=1}^{N} t(a_i)+\mathop{}\limits_{i=0}^{N}\tau_D(a_i,a_{i+1}) \qquad (1)$$

$$t(w_{par}(a_1,a_2,\ldots,a_N))=\max_{i=1}^{N}(\tau_D(a_0,a_i)+t(a_i)+\tau_D(a_i,a_{N+1})) \qquad (2)$$

where $t(a_i)$ is the execution time of the ith operation, $\tau_D(a_{i-1},a_i) = d(a_{i-1},a_i) \quad s(a_{i-1},a_i)$ is the time for data transfer between a_{i-1} and a_i, $d(a_{i-1},a_i)$ is the volume of data being transferred from a_{i-1} to a_i operation, and $s(a_{i-1},a_i)$ is the speed of data transmission channel (network bandwidth between the nodes where opertaions a_{i-1} and a_i are running). It is also assumed that data should also be delivered to the first operation ($d(a_0,a_1)$) and fetched from the last one ($d(a_N,a_{N+1})$). As soon as the most of the real-world workflow examples can be fully or partially represented as the hierarchical composition of sequence and parallel execution flows (e.g., $w_{seq}(a_1,w_{par}(w_{seq}(a_2,a_3),a_4),a_5)$ at fig.1) so this estimation approach has more than only theoretical value.

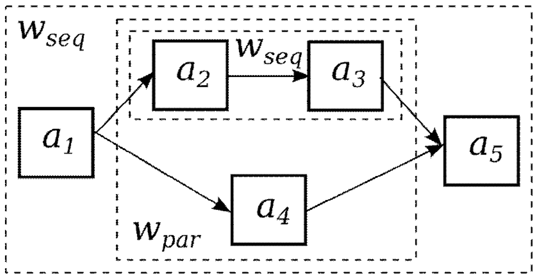

Fig. 1. Workflow example as the combination of sequential and concurrent branches

Aside from direct estimation of workflow execution duration it can be convenient to evaluate the part of overall time spent for useful calculations t^+ related to the overall execution time t:

$$K_t^+(w_{seq}(a_1,a_2,\ldots,a_N)) = \frac{\sum_{i=1}^{N} t^+(a_i)}{t(w_{seq})} = \frac{\sum_{i=1}^{N} t^+(a_i)}{\sum_{i=1}^{N} t(a_i) + \sum_{i=0}^{N} \tau_D(a_i,a_{i+1})} \tag{3}$$

$$K_t^+(w_{par}(a_1,a_2,\ldots,a_N)) = \frac{\sum_{i=1}^{N} t^+(a_i)}{\sum_{i=1}^{N} (\tau_D(a_0,a_i) + t(a_i) + \tau_D(a_i,a_{N+1}))} \tag{4}$$

The opposite quantity characterize the part of overhead for data transfer, queue delays and other side actions or being idle while workflow execution:

$$K_t^-(w) = 1 - K_t^+(w) \tag{5}$$

The estimation of **workflow operation execution time** $t(a)$ is a key part of the overall workflow execution time estimation. Prediction of the operation execution time is a non-trivial procedure that should consider a set of factors often stochastic or those hard to formalize:

$$t(a) = \sum_i t_{Q_i}(a) + \sum_j t_{I_j}(a) + \tau(a) + \delta(a) + \sum_k t_{O_k}(a) \tag{6}$$

where t_{Q_i} stands for waiting time of computation request in ith queue of queue system (e.g. this can be scheduler or LRMS queues for grid jobs). The stochastic quantity that generally depends on the current load of the target resources and internal scheduling algorithms; t_{I_j}, t_{O_k} stand for the time needed to parse input data and compose output data for jth (kth) level protocol (e.g. HTTP(S) and SOAP for web services). Depend on the overall data size, the size of useful data load, data format of the protocol and the way data is stored (encoding, encryption), resource performance; τ stand for pure time for useful computations. Depends on the algorithms implemented in program (e.g. time complexity), data volumes to process, resource performance; δ describes delays connected with the supporting procedures (program loading, initialization, establishing network connection, interacting with file system and other OS or middleware (M/W) resources etc.).

Workflow execution levels

Among the WfMS components there are two main kinds of entities [3]: worklfow operations and their execution manager. The manager is responsible for execution of operations in prescribed order. This component is only available for the centralized approach to workflow execution as opposed to decentralized scenario when operations are responsible for invocations of each other by themselves.

Local level of execution means performing computations on the local user's machine with its limited computing power. It can be taken for simplicity that $t_i(a) = 0$, $t_{I_j}(a) = t_{O_j}(a) = 0, j > 1$ (no intermediate protocols and data representations usually needed), $\delta(a) \to 0$ (no considerable delays while interacting wih local OS), $\tau_D(a_i, a_{i+1})$ depends on data exchange organization: via RAM or file system.

Resource level of computations means the dedicated high performance resource (computer cluster as an example) with its own computing control system named local resource management system (LRMS) consisting of a scheduler and resource manager. LRMS represents computer cluster as the single abstract resource where computations can be launched on any free node from its pool. But in case of full load of all nodes computing task may wait in the queue for an unpredictable time.

At this level the following should be considered additionally: stochastic wait time in LRMS queue $t_{Q(LRMS)}(a)$, additional overhead for data transfer to resource and back through the network with speed $s_R(a_i, a_{i+1})$, as well as for internal data transfers between nodes with speed $s_N(a_i, a_{i+1})$ and overhead δ_{LRMS} for interaction with LRMS.

Grid computing technology as the way of organizing the shared access to remote resources is not only the trade-off solution for dedicated resource maintenance expenses, but also provides the reserved reliability in case of breakdowns of some resources. Grid level is an additional abstraction level for a set of computer clusters and other types of resources representing them as the single virtual supercomputer with access granted according to the user's membership in the virtual organization allowed to utilize those aggregated resources. The price for such uniform user access to resource pool is an additional grid middleware (M/W) layer organizing the interaction with grid resources (e.g. submitting grid computing job via the grid scheduler).

At this level the following should be considered additionally: stochastic waiting time in grid scheduler queue $t_{Q(G)}(a)$, as well as an additional overhead for possible internal data transfers between resources via the netrowk at speed $s_{G_j}(a_i, a_{i+1})$ and overhead δ_G connected with M/W operation.

Service level means execution of web services. They can serve as the universal public platform-independent standardized interface for workflow operations execution, hiding the internal details of their implementation and execution parameters, exact location of program code thus serving well as remote access wrappers for local, cluster or grid computations.

At this level the following should be considered additionally: data transfer overhead through the network at speed $s_S(a_i, a_{i+1})$ (in case of centralized orchestration there is additional data transfer toward and back from the orchestrator at the speed $s_O(a_0, a_i)$) as well as significant SOAP messages parsing overhead $t_{I(SOAP)}(a_i)$ and $t_{O(SOAP)}(a_i)$ for input and output respectively.

Workflow management implementation options

Before proceeding to the practical feasibility analysis of different approached to SOA WfMS implementation let's define typical test environment parameters (based on practical experience). For simplicity let's assume that s = 100 Mbit/s for any network transfer, δ_G = 1 min (all delays related to M/W including target resource discovery and grid informational system refresh interval, also all delays connected to LRMS δ_{LRMS} are included for convenience), $t_{I(SOAP)} = t_{O(SOAP)} = d(a_i, a_{i+1}) \quad s_P$ where s_P = 100 MB/s is an average speed of SOAP message parsing.

1. Level of workflow decomposition. Many of widely used local WfMS [3] allow creation of workflows composed from fine-grained operations, in some sense they are similar to visual programming tools. This approach provide maximum flexibility but requires necessary qualification of end user. In such systems any real-world task would be represented as the workflow with a lot of atomic operations and high complexity thus leading to considerable overheads and due to numerous data transfers so overall execution time $t(w)$ would increase dramatically and often unacceptably.

Let's consider typical task of N-sized SLAE solving with LU decomposition method. If we construct computation procedure as the workflow consisting from only primitive arithmetic operations (like in block diagram of the algorithm) then the number of invocations can be estimated as $O(N^3)$. When compared to monolithic program it gives near N^3 additional delays for data transfers. In case of workflow implementation at service level even with neglectfully small data volumes being transferred each time network and service invocation delays will be near $\delta(a_i)$ = 10..100 ms (and $\delta(a_i) \quad \tau(a_i)$) thus giving approximately $10^7 \dots 10^8$ additional seconds of execution for relatively small matrices of $N = 10^3$.

If we implement only coarse-grained stages of LU-factorization (a_1) and backward substitution (a_2) as operations then the transfer overhead for data size $d(a_1,a_2) = kN^2$ (where k is the size of each number in SOAP envelope, approx. 10 bytes) will be $\tau_D(a_1,a_2) + t_{O(SOAP)}(a_1) + t_{I(SOAP)}(a_2) \approx$ 1 sec, quite affordable for end user. And it is also remains possible to utilize LU – factorization service separately.

These time estimations will double for centralized execution (when data is not transferred directly but via the controller like BPEL orchestrator server for web service level).

This artificial example illustrate some important conclusions: 1) composition of workflows with high level of operations decomposition provide more freedom for engineer but requires him to have programmer skills; 2) execution of workflows with a lot of operations and/or intensively repeated loops should be localized when possible; 3) being highly composable the basic primitive operations on numeric and text data can be included into WfMS as services for supporting usage (e.g. to connect coarse-grained operations when it cannot affect $K_t^+(w)$ significantly).

2. Workflow with queued operations. Not only in case when workflow is composed from remote components (like web services) but also when its operations can be queued (for resource and grid levels this can sometimes lead to unpredictably long delays) the level of decomposition should be such that minimize the sequential data transfers. Thus resource limitations overcoming leads to severe restrictions on workflow configuration. But even in such case there can be additional benefits.

1) Eq. (2) implies that execution time does not generally depend on the number of parallel branches. This allows to reach time acceleration compared to local level execution even in case of significant overhead $t^-(a) = t(a) - \tau(a)$. Such acceleration can be possible if $t(w_{par}^R) > t(w_{par}^L)$ (where R index is for remote execution, L is for local) or:

$$\max_{i=1}^{N}(\tau_D(a_0,a_i^R) + t(a_i^R) + \tau_D(a_i^R,a_{N+1})) < \max_{i=1}^{N} t(a_i^L) \quad (7)$$

which can happen in case of local resources deficit. In case of N equivalent operations $a_i = a \mid i = 1..N$ (e.g. for tasks where input data can be easily distributed) remote execution will benefit if:

$$\max_{i=1}^{N}\left(\tau_D(a_0,a_i^R) + \tau_D(a_i^R,a_{N+1}) + {}_{j}\, t_{I_j}(a_i^R) + {}_{k}\, t_{O_k}(a_i^R) + {}_{q}\, t_{Q_q}(a_i^R) + \delta(a_i^R) + \tau(a_i^R)\right) < \tau(a^L) \quad (8)$$

For example, in case of grid execution with C_G available cores same as $C_L < N$ local ones then (considering grid scheduler queue and ignoring input and output data preprocessing) grid execution is preferable if:

$$\tau_D(a_0,a) + \tau_D(a,a_{N+1}) + \left\lceil \frac{N}{C_G} \right\rceil (\delta_G(a) + \tau(a)) < \frac{N}{C_L}\tau(a) \quad (9)$$

2) The absence of guaranteed time acceleration can be compensated with possibility of parallel execution of alternative branches solving the same task with different methods which can increase overall reliability (probability of positive outcome).

3) Remote computing can also benefit in providing the opportunity of access to the unique software resource which is impossible to integrate in system directly (due to lack of source codes or license etc.).

3. Multi-level workflows. In order to cope with local resource limitations it can be reasonable to implement the multi-level workflow model when coarse-grained operations at the service level initiate and possibly control the computations at local, resource or grid level. This trade-off solution will give the opportunity to overcome resource restrictions and avoid intensive overheads while retaining the certain level of flexibility for workflow composition by the end user (fig.2).

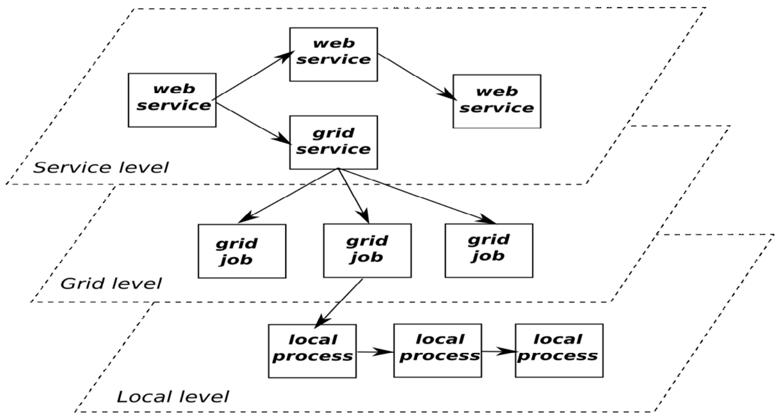

Fig. 2. Multi-level workflow concept

This multi-level approach was successfully used to integrate existing tools into an interdisciplinary modeling software during development of the Web / Grid ALLTED [1] simulation complex under the project "Interdisciplinary complex of optimal mathematical modeling in grid environment with the automatic composition and solving of equations of corresponding mathematical models".

Conclusion

In this paper the application of workflow methodology for execution of complex user customizable computing scenarios in service-oriented CAE/CAD software was studied. Service-oriented architecture has following benefits for engineering tools:

- loose-coupled services with standardized open interfaces simplify independent development, distributed deployment of functionality as well as integration of third-party tools;
- remotely deployed services can provide access to grid computing resources to eliminate local computing resource deficit for compute-intensive tasks;
- SOA allows dynamic coupling of services (orchestration) that can be used to organize workflows of computations according to user scenarios.

General approach to time estimation of workflows organized at different architecture levels was covered. The other side of SOA flexibility is communication overhead between remote services so recommendations on minimization of this penalty were provided. Multi-level workflow concept combining advantages of local, grid and service-level execution was also described.

1. Zgurovsky M. WebALLTED: Interdisciplinary Simulator Based on Grid Services / Zgurovsky M., Petrenko A., Ladogubets V., Finogenov O., Bulakh B. // East-West Design & Test Symposium: 10-th IEEE East-West Design & Test Symposium «EWDTS'2012», 14-17 September 2012, Kharkov, Ukraine: proc. – Kharkov, 2012. – P. 126–129. 2. Erl T. Service-Oriented Architecture: Concepts, Technology & Design. – New York: Prentice Hall / PearsonPTR. – 2005. – 792 p. 3. Workflows for e-Science. Scientific Workflows for Grids / I.J. Taylor, E. Deelman, D.B. Gannon, M. Shields (Eds.). – Springer. – 2007. – 530 p.