Decentralized Service Orchestration by Continuous Message Passing

R.JAYAPRAKASH, M. SHANMUGAM , P.MANIKANDAN, and S. SHIVARAJ
Department of Computer Science
Pondicherry University
Puducherry, India.

Abstract— **Decentralized service stands for distributing segments of workflow among various workflow engines, and workflow has set of activities responsible for invoking Web Services. Decentralized Orchestration holds an upper hand over Centralized Orchestration in producing optimal solutions in terms of scalability and network traffic by overcoming inefficient routing of messages. Although decentralized orchestration being more optimal in scalability it lacks in very significant part of web services and internet called reliability. So, we found a way out to solve this issue by introducing a reliability phenomenon called Continuous Message Passing. This Continuous Message passing on merging with Decentralized Orchestration is believed to deliver the most trusted web service and produce optimal solutions. The combination of these two not only optimizes reliability to greater extent but also produces substantial increase in scalability which assures workflow enhancement. With communication and processing being integral part of web services, the model framed believed to convey greater enhancement in both communication and processing of resources which would make best use of workflow of Web services and Internet.**

Keywords- Decentralized Orchestration, Reliability, Continuous Message Passing, Workflow, Web Services

I. INTRODUCTION

A web service is defined as “a software system designed to support interoperable machine-to-machine interaction over a network”. Web Service-oriented computing provides a programming pattern for achieving and improving business process. Business software assists business process to emerge. Efficient business software relies on interoperability, which can be achieved through Enterprise Application Integration (EAI).

Web services are one of the processes to attain EAI. Web Service Description Language (WSDL) that is capable of being accessed through standard network protocol is best related to service that describes Web service and it can be accessed via protocols such as SOAP over HTTP with an XML serialization and other standards. We talked about achieving and improving business process, it can be achieved by execution of workflow to invoke web service. Different web service providers provide different web services and it carries workflow which performs activities in sequence or parallel. The workflow in web service is of two types, Composite and Basic.

Basic web service is a web service that does not invoke other web service where as composite web service invoke one or more web services for combined processing. This coordinated process of invoking a web services is achieved through Orchestration. Web service Orchestration controls the order of the condition for the web services invocation. Orchestration web service invokes one or more web services from different service provider in coordinated manner and orchestration decides whether or not to invoke the web services.

The Workflow execution can be achieved by Centralized approach where workflow execution is carried out by only one workflow engine placed on single server. This centralized approach is not optimal in the process of controlling the correct and reliable execution of activities, since it has to bare a larger amount of resources on single engine. So centralized approach faces huge problem in terms of communication and processing which are core activities for business process.

Centralized workflow approach can be resolved by allocating different segments of workflow among various workflow engines. This way of approach is termed as Decentralized workflow. In decentralized approach each workflow engine communicates with other workflow engines with out depending on single workflow engine. This approach increase scalability to reasonable extent but this requires pre-allocation of resources. With pre-allocation some resources remain unused even after workflow execution has been completed. So by pre-allocation the unused resources are blocked, which otherwise might have come handy for other purposes. This definitely is a big set back in terms of resource allocation were it reflects negative effect on scalability.

Communication and processing through continuous message passing is again a decentralized approach through workflow enactment. Integrating continuous message passing with decentralized approach would pre-allocate the resource prior to the workflow execution and release the allocated resource after execution is completed. This is only achieved by interaction among the resources through continuous message passing. Continuous message passing approach provides optimal use of resources that speeds up communication and processing.

II. RELATED WORK

When web services started emerging in the internet scenario the most important feature called composition techniques too started to get its phenomenal growth by many contributors thought, by the way there are some major composition techniques like Orchestration and Choreography [1] that took web service composition to achieve successful result. Even there was some research that went under the environment that composition taking place. There the centralized approach and decentralized approach both have produced the result in optimal success and even have their own drawback within.

Previously by a set of Input and output parameters some experiments are undergone to clarify the best suite composition technique using the decentralized workflow enactment, in that research most of the results witnessed that Decentralized Service Orchestration [2] sounds more than the Centralized one in all aspects in especially overcoming the most important issue called bottleneck problem and traffic over network, but still some drawbacks that pulls decentralized back like communication between the services that involved in the composition. To execute the services with efficient way Triggers are maintained in each web services and they used to invoke and build the desired output. In this way of composition the most expected limitation called enhancing triggers took place and needed the additional burden to the developers to involve the triggers in the each service which doesn't make feasible to all the global web service developers.

When web services are increased in large scale sure it going to dominate the software industries very soon, when composition techniques of these web services are reliable and makes the user's be comfort in the consuming and get expected result it would lead to achieve the expected success ratio. From various self-contained algorithm, some algorithms are mostly based on the current existing standards. In [8], a BPEL-based Web service composition using high-level Petri-Nets (HPN) approach is proposed.

III. DECENTRALIZED WORKFLOW ORCHESTRATION

Decentralized workflow is derived from Centralized workflow. In a centralized workflow, workflow engine is placed on a single server which is responsible in controlling entire workflow activities. Model has been framed to show the workflow execution is illustrated in figure: In the figure sequence of three work items are named as A, B and C for workflow execution. Here single workflow engine handles all the three resource a, b, and c by sending and receiving the work item before and after processing. Resource 'a' after completing the process sends back the result to workflow engine, only then work item for the resource 'b' is sent. So each and every resource has to wait for the other resource to complete their process and get result.

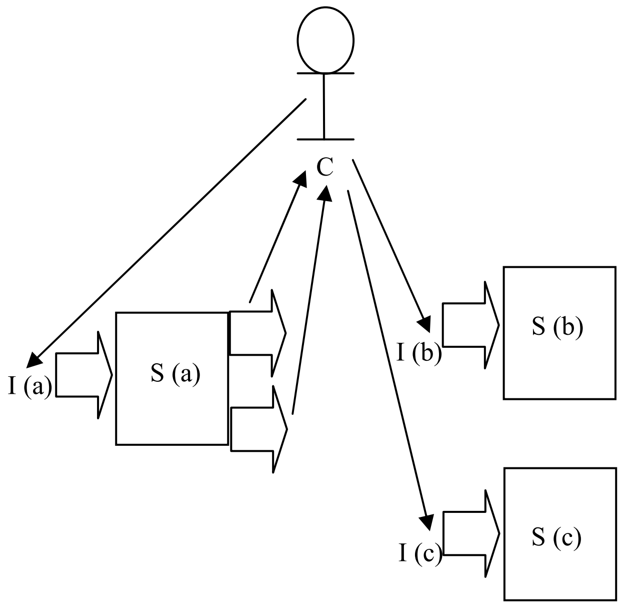


Figure 1: workflow execution

So the problem with this centralized workflow approach is communication and processing which is an integral part of failure.

In the Decentralized approach the resources are pre-allocated. Decentralized approach model has been illustrated in figure Here work items a, b, c are handled by three different work engines namely w1, w2 and w3. Each work engine is dedicated to its work item.

Where work engine 'w1' is dedicated to work item 'a', work engine 'w2' is dedicated to work item 'b', and w3 is dedicated to 'c'. This approach definitely gives faster execution but problem arises at resource pre-allocation. All the work items are pre-allocated by the work engines before the execution. So all the resources that are necessary for the workflow execution are allocated. If this resources are not blocked it would have been used for other purpose.

IV. CONTINUOUS MESSAGE PASSING

Continuous message passing is again a decentralized approach through workflow enactment. Continuous message passing would pre-allocate the resource prior to the workflow execution and release the allocated resource after execution is completed.

In the Continuous Message Passing approach the processed resources would be freed for other purpose. Continuous Message Passing model has been illustrated in figure 2. Here work items a, b, c are handled by three different work engines namely w1, w2 and w3.

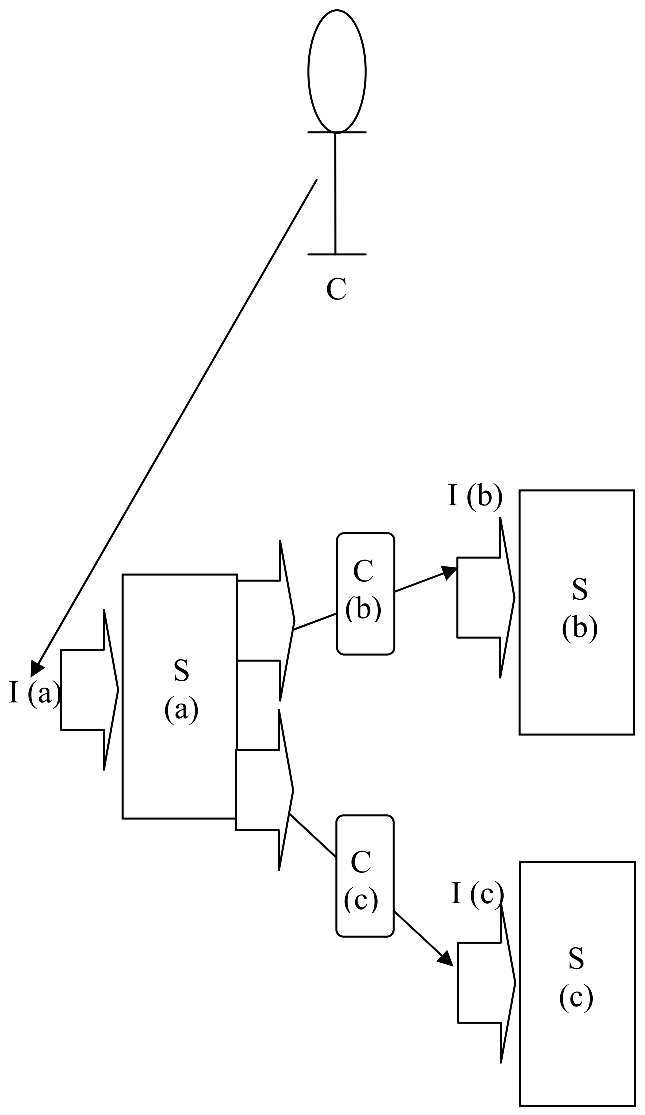


Figure 2: Continuous Message Passing model

Workflow engine w1 after processing resource 'a' the resource is released. Similarly workflow engine w2 and w3 after processing resource 'b' and 'c' release it. So the overall work shows that after one workflow engine has completed its work, then continuation is passed on to the next workflow engine. This approach definitely gives faster execution and increases scalability and reliability.

V. WORKFLOW

The key abstraction in the workflow model is flow, in which the term flow corresponds to workflow. Flow is defined as follows,

Flow::== Empty Flow | Activity | Blackbox Flow | seq(Flow*) | fork(join-agent, Flow*) | or(Flow*) | if(Condition, Flow, Flow) | loop(Condition, Flow*)

The flow deals three more important flows,

Empty Flow: Empty Flow is used to indicate the completion of the execution.

Activity: Activity is an agent that is responsible for its execution of the program, composition of program, the input and output data, etc.

Black-box Flow: Blackbox Flow is a flow whose internal structure is only known by its own agent, and it can be combined by a central workflow engine.

Some of the more complicated flows can be described as follows:

seq: seq defines a sequence of sub-flows.

fork: fork spawns multiple parallel branches of sub-flows. Join agent combines parallel branches, and it can be automatically picked during the time of execution of a workflow.

or: or enables execution from multiple alternative sub-flows.

if: if chooses a flow by determining if a condition is true or false. A condition is defined on either flow-relevant data or execution status of the flow.

loop: Loop defines a sub-flow that is repeated until the condition is evaluated to be false.

An example of a flow specification can look like the following:

seq(Aa, fork(e, or(Bb, Cc), Dd), Ee)

Aa, Bb, Cc, Dd and Ee denote activities to be executed by agents at the sites a, b, c, d and e, respectively. A compensating activity for Aa would then look like Aa^{-1}.

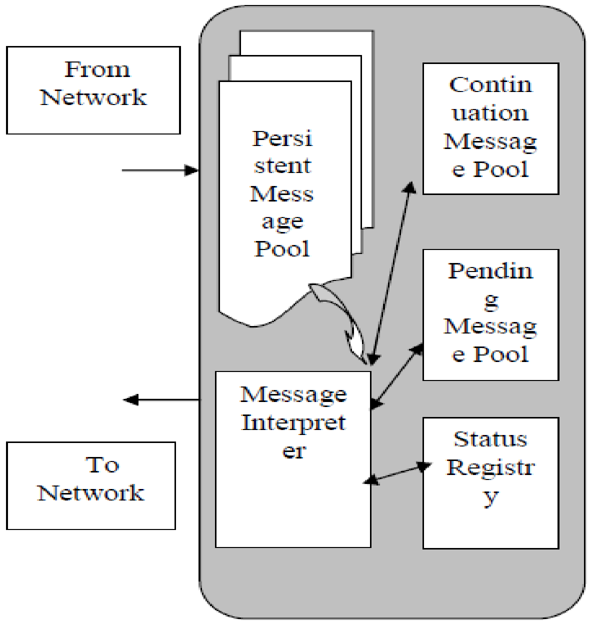


Figure 3: Structure of Message Container

VI. Performance Results

CEKK is an abstract state machine for distributed and recoverable flow enactment, where C, E and K represent control, environment and continuation, respectively [13]. According to [13], the CEKK machine can be looked at from two angles:

Local CEKK machine: A local CEKK machine defines possible states and their possible transitions locally at an agent.

Global CEKK machine: A global CEKK machine defines the possible global states of a flow and the possible transitions among them. It consists of a number of local CEKK machines. The global state of the flow is the aggregation of the local states and the global state transitions are defined solely by the local state transitions.

A state of a local CEKK machine at agent p is a quadruple < c; e; ks; kf >p, where c is called a control expression, e an environment, ks is called a success continuation and kf is called a failure continuation [13]. According to [13], they can be defined as follows:

Control expression: control expression represents the next flow to be enacted immediately.

Environment: environment is the runtime context of the flow.

Success continuation: success continuation is the continuation towards the successful end of the flow.

Failure continuation: failure continuation is the continuation towards the compensated end of the flow.

Transition rules: According to [13], the state transition appears in one of four forms as follows:

Local ongoing: Local state transition performed by agent p.

$$< c_0, e_0, ks_0, kf_0 >_p \rightarrow < c1, e1, ks1, kf_1 >_p$$

Remote forwarding: Send state information from agent p to agent q.

$$< c, e, ks, kf >_p \rightarrow < c, e, ks, kf >_q$$

Local divergence: Agent p spawns multiple parallel branches, where each branch is independent from the other branches.

$$< c_0, e_0, ks_0, kf_0 >p \rightarrow$$

$$\{<c_1, e_1, ks_1, kf_1 >_p, < c_2, e_2, ks_2, kf_2 >_p, \ldots, < c_n, e_n, ks_n, kf_n >_p\}$$

Local convergence: Agent p joins spawned branches.

$$\{< c_1, e_1, ks_1, kf_1 >_p, < c_2, e_2, ks_2, kf_2 >_p, \ldots, <c_n, e_n, ks_n, kf_n >_p\} \rightarrow < c_u, e_u, ks_u, kf_u >_p$$

Remote forwarding is asynchronous message passing between agents. In all other cases, state transitions are carried out locally at individual agents. This explains why global co-ordination is not needed among the agents [13].

The names of the transition rules have insight meaning. For example, the transition rules named A1 and S1 refers to an activity and a sequence, respectively. According to [13], the transition rules named A1, A2, A3, S1 and S2 can be listed as follows:

A1: A1 represents an agent q that is sending state information to another agent p.

$$< A_p, e, ks, kf >_q \rightarrow < A_p, e, ks, kf >_p \text{ if } p \neq q$$

A2: A2 represents a successful execution of an activity at an agent p where a compensation activity is added to the failure continuation.

$$< A_p, e, ks, kf >p \rightarrow < ks{:}head, succ\ (A_p) : e, ks{:}tail, A^{-1}_p : kf >_p \text{ if } succ(A_p)$$

A3: A3 represents a failed execution of an activity at an agent p, which means that the failure continuation is enacted.

$$< A_p, e, ks, kf >_p < kf{:}head, fail\ (A_p) : e, kf{:}tail, kf >_p \text{ if } fail(A_p)$$

S1: S1 represents the last sub-flow to be enacted in a flow sequence, which means that nothing is pushed to the success continuation.

$$seq(f_s), e, ks, kf >_p \rightarrow < fs{:}head, e, ks, kf >_p \text{ if } |f_s| = 1$$

S2: S2 represents a flow sequence with multiple sub-flows, which means that the first sub-flow is enacted. The other sub-ows are pushed to the success continuation.

$$< seq(f_s), e, ks, kf >p \rightarrow < fs{:}head, e, seq\ (fs{:}tail) : ks, kf >_p \text{ if } |fs| \neq 1$$

The next transition rules named F1 and J1 are somewhat complicated, therefore a state initially represented as < c, e, ks, kf > will be shortened to < c, ks >. A reason for doing this is because recoverable flow enactment is not a goal for this thesis. The modified transition rules can be defined as follows:

F1: F1 represents agent p which spawns multiple parallel branches.

$$< fork(q, f_1, f_2, : : : , f_n), ks >p \rightarrow$$

$$\{< f_1, join\_succ , ks >p, < f2, join\_succ : ks >_p, \ldots, < f_n, join\_succ : ks >_p\}$$

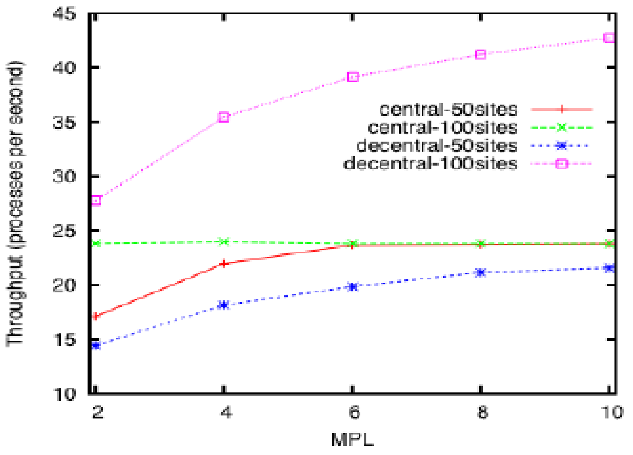
Where

join_succ = join(q, and(succ(f_1), succ(f_2), : : : , succ(f_n)))

J1: J1 represents joining spawned branches.

{< join_succ, ks $>_q$,….,< join_succ, ks $>_q$} → < ks:head, ks:tail $>_q$

The transition rules A1, A2, A3, S1 and S2 are valid for an abstract CEKK engine, while the transition rules F1 and J1 are valid for an abstract CK engine, where C stands for control and K stands for continuation.



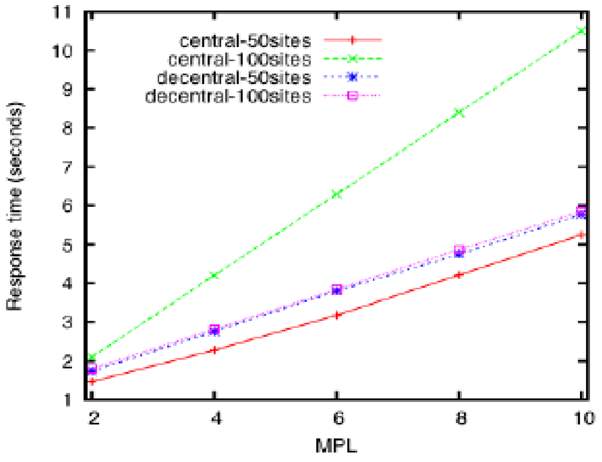


Figure 4: Performance Evaluation

The performance evaluation of 50 and 100 service sites for processes with the 4x4 structure is shown in figure 4. When there are 50 service sites, the centralized approach outperforms the continuation-passing one, both in total throughput and in process response time. However, when there are 100 service sites, the central engine gets congested, whereas with continuation passing, the system throughput still grows with the increase of the system load.

VII. CONCLUSION

Main Objective of our proposed work is to overcome the listed drawbacks of the existing centralized and decentralized approach, even they sounds well in flexibility and scalability they stands a step back in the aspects of reliability factor. Our approach that to contribute in the aspect of reliability that makes the co-ordination between the services and provides the resultant value to the client. In Continuation message passing the messages carry the continuation result, it is also called the partial result or intermediate result, and these results are orchestrated to produce desired output. All the above the resource pre-allocation problem completely vanished and dynamic allocation of the resource comes into picture to provide the ultimate scalability.

REFERENCES

[1] Adam Barker, Christopher D. Walton, and David Robertson, "Choreographing Web Services", In *IEEE TRANSACTIONS ON SERVICES COMPUTING*, VOL. 2, NO. 2, APRIL-JUNE 2009.

[2] Walter Binder, Ion Constantinescu, Boi Faltings, "Decentralized Orchestration of Composite Web Services" In *IEEE International Conference on Web Services (ICWS'06).*

[3] Nancy Cova Suazo, José Oscar Olmedo Aguirre, "Aspect-Oriented Web Services Orchestration", In 2nd International Conference on Electrical and Electronics Engineering (ICEEE) and XI Conference on Electrical Engineering (CIE 2005).

[4] Daniel Martin Daniel Wutke Frank Leymann, "A Novel Approach to Decentralized Workflow Enactment" In *12th International IEEE Enterprise Distributed Object Computing Conference*.

[5] Ustun Yildiz and Claude Godart, "Information Flow Control with Decentralized Service Compositions" In 2007 *IEEE International Conference on Web Services (ICWS 2007).*

[6] Saayan Mitra, Ratnesh Kumar, Samik Basu "Optimum Decentralized Choreography for Web Services Composition" In 2008 *IEEE International Conference on Services Computing*.

[7] Walter Binder, Ion Constantinescu, Boi Faltings "Decentralized Orchestration of Composite Web Services" In *IEEE International Conference on Web Services (ICWS'06).*

[8] W.-L. Dong, H. Yu, and Y.-B. Zhang. "Testing bpel-based web service composition using high-level Petri nets". In *Proc. of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC '06)*, pages 441–444, Oct. 2006.

[9] Esin Gokkoca, Mehmet Altinel, Ibrahim Cingil, E.Nesime Tatbul, Pinar Koksal, Asuman Dogac "Design and Implementation of a Distributed Workflow Enactment Service".

[10] Walter Binder, Ion Constantinescu, Boi Faltings "Service Invocation Triggers: A Lightweight Routing Infrastructure for Decentralized Workflow Orchestration" Proceedings of the *20th International Conference on Advanced Information Networking and Applications (AINA'06).*

[11] Ustun YILDIZ and Claude GODART "Synchronization Solutions for Decentralized Service Orchestrations" *Second International Conference on Internet and Web Applications and Services (ICIW'07)*

[12] Zaharina Velikova, Julian Schütte, Nicolai Kuntze "Security in Decentralized Workflows"

[13] Yu and Yang. "Continuation-passing enactment of distributed recoverable workflows". In *Proceedings of ACM SAC'07*.

[14] Weihai Yu "Scalable Services Orchestration with Continuation-Passing Messaging" 2009 *First International Conference on Intensive Applications and Services*.

[15] Yajuan Song , Lei Liu ,Ping Ren "Web Service Composition Based on the Annotated Ontology" 2009 *First International Workshop on Education Technology and Computer Science*.

[16] Mark Wiley Aihua Wu Jianwen Su WSDL-D: "A Flexible Web Service Invocation Mechanism for Large Datasets" 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services.

[17] Mohamad EL Falou, Maroua Bouzid, Abdel-Illah Mouaddib ,Thierry Vidal "Automated Web Services Composition : A Decentralised Multi-Agent Approach" 2009 *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology – Workshops*.

[18] Ustun Yildiz and Claude Godart "Centralized versus Decentralized Conversation-based Orchestrations" The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007).