VOL. 16, NO. 4, FEBRUARY 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

APPLICATION OF SERVICE-ORIENTED COMPUTING IN COMPUTATIONAL SCIENCE: WAVE PROPAGATION AND SCATTERING IN VEGETATION

Michael Yu-Chi Wu
Department of Management of Information System, College of Business, University of Houston-Clear Lake, TX
E-Mail: wum@uhcl.edu

ABSTRACT

In the world where collaboration among multidisciplinary services is necessary for continuous improvements, scientific computing, unfortunately, lacks the infrastructure that allows for this type of collaboration. As such, this paper recommends guidelines of service-oriented, multi-layered designs through the exemplifications of models, service-oriented architecture, architectural layers, service compositions, and computational stages. The objective is to apply these services-computing principles to the designs of scientific-computing applications. To establish the validity of these suggestions, further benchmarks that run on 32 computers have demonstrated consistent speedups of at least 20 times with negligible latencies in service orchestrations (under 1% of the total simulated time). This study provides sciences with procedures that coalesce with technologies through services computing platforms.

Keywords: distributed computing, radiative transfer, services computing, services orchestration and composition, service-oriented architecture, workflow.

Article Highlights

- Describe the benefit of utilizing services-oriented computing in a computationally intensive problem: wave propagation and scattering in vegetation, which plays an integral role in accurately calculating remote sensing, an application that is frequently used in the study of global warming, deforestation monitoring, agricultural field resource management and conservation.
- Illustrate and develop service designs for the problem of interest in the construction of a parallel simulator.
- Propose, through the design of the parallel simulator, how to integrate services-computing strategies into a service-oriented computing application.

INTRODUCTION

Service-oriented computing, or services computing for short, has become an integral part of information technology through a service-oriented relation between software solutions and business processes (Bichier and Lin 2006; Papazoglou *et al*. 2007). The service-oriented architecture (SoA)-a style of software design in support of services-allows each service to discover and become aware of other services on the network, thereby laying the foundation of services computing (Huhns and Singh 2005; Chen *et al*. 2014). In principle, services computing ideally combines the fields of sciences and technologies through computational abstractions, architectures, techniques, and tools to transition between scientific services and information technology (IT) services (Bouguettaya *et al.* 2017). However, scientific computing continues to solve computational problems by using traditional, high-performance-computing frameworks through low-level, rudimentary computer networks, storage devices, and software libraries (such as the discussions (Shainer *et al*., n.d.) and (Simmendinger *et al.* 2019)), rather than fully utilizing service-oriented computing, as in the fields of technology and business (Bouguettaya *et al*. 2017). Some areas of computational sciences may require high-speed computing powers (AbdelBaky *et al*. 2012) while others may utilize large datasets (Rodriguez and Buyya 2014), therefore have been utilizing basic algorithmic, optimal, and infrastructural designs for rapid calculations (e.g., (Slotnick *et al.* 2014)). These designs are single use only and must be written by scientists, who often lack the expertise in computer coding (Storer 2017). As a result, they are usually suited to solving a single scientific problem, rather than being dynamically applicable to a broader range of problems (e.g., (Keyes *et al*. 2013)).

The novelty of this paper's proposal is further highlighted by the lack of progress in the field of science computing since the 1940's: science is no longer an isolated field with scientists working in solitude. Instead, they are becoming increasingly reliant on software to generate pertinent data and results. According to Storer (2017), scientists having no coding training frequently write their own software, which are ineffective due to lack of reproducibility of data generated or software validation. This problem is further compounded by the fact that scientists are unaware of the most updated software tools. Published scientific findings need to be retracted, invalidating conclusions drawn from such data. The absence of a seamless integration of service computing with scientific computing is stifling advances and progress in science. Science should not fall behind to other fields, especially when these other fields-business, healthcare, financial management, and tourism (AbdelBaky *et al.* 2012) have already been utilizing services computing.

This paper demonstrates how to incorporate service-oriented methods by way of solving the

computationally challenging calculations of electromagnetic wave propagation in vegetation. This computation has applications in real-time approximation of remote sensing, which is frequently used in the study of global warming, deforestation monitoring, agricultural field resource management and conservation.

PROBLEM MODELING: APPLICATION OF THE SERVICES COMPUTING MODEL

This study proposes the construction of a simulated model (Wu, Lin, and Sun 2017) utilizing service-oriented computing (Figure-2) to solve a remote-sensing problem of how electromagnetic waves propagate and scatter in vegetation (i.e., random media, as illustrated in Figure-1) due to an external source that radiates incident-pulse trains of electromagnetic waves. These waves can be plane waves (i.e., sunlight) (Whitman *et al.* 1996), collimated-beam waves (i.e., laser beams) (Whitman, Schwering, and Wu 2007), and spherical-beam waves (i.e., antenna radiations) (Whitman, Wu, and Schwering 2010). Appendix A provides a summary of the derivation for the solution to this problem.

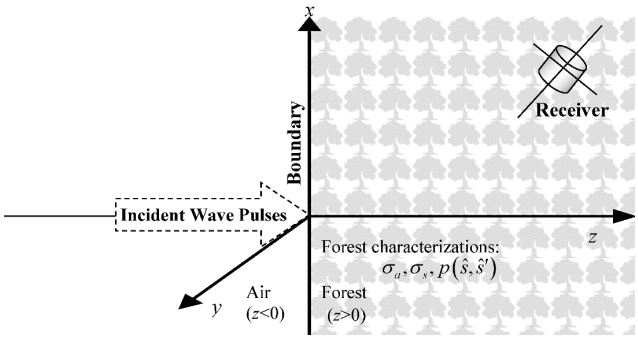

Figure-1. This problem evaluates the power at a receiver placed somewhere in the right-half region of the vegetation-a random medium characterized by the parameters σ_a, σ_s, and $p(\hat{s}, \hat{s}')$. The receiver gathers the electromagnetic waves, radiated from outside (i.e, left-half region) of the vegetation, while the objects in the vegetation, including branches and leaves, are dispersing and scattering these traveling waves.

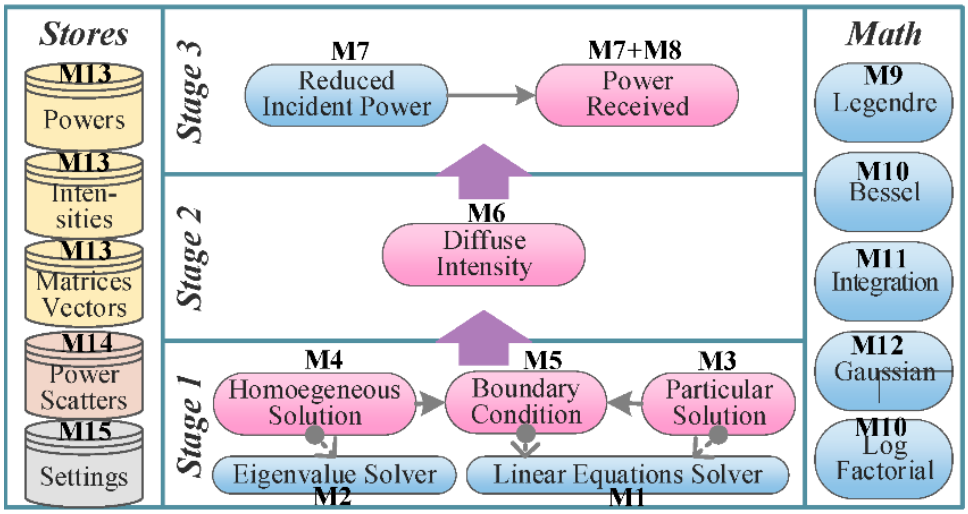

Figure-2. The services-computing model layer consists of five layers: store and math in addition to stage 1 (matrices), stage 2 (intensities), and stage 3 (powers) computations. All the stages are making use of the math and the data stores; see models in Table-1.

Figure-2 is the schematic representation of a Service Computing Model in order to solve the wave scattering problem. The author has categorized mathematical functions required to solve the wave propagation problem into independently functioning levels, designated as "Stage 1", "Stage 2," and "Stage 3." Each level consists of mathematical equations. In the end, the application/simulator is expected to calculate the power received at the receiver placed inside the vegetation.

Table-1. This table summarizes all the models with their dependencies, asdelineated in the Services Computing in Figure-2, within their own functioning layers: Stage 1, Stage 2, Stage 3, Mathematics, and Stores.

	MODEL	CALCULATING	WITH
Stage 1	M1: Linear Solver	System of linear equations, $A \times x = b$ with matrix **A** and vector b, along with preconditioners and iterative refinements	
	M2: Eigenvalue Solver	Generalized eigenvalue system of linear equations, $A \times V = \Lambda \times B \times V$ with matrices A and B, where Λ are the eigenvalues and V are the eigenvectors.	
	M3: Particular Solution	The vector F in the linear systems of particular equations $B_0 \times F = f$	M1
	M4: Homo-geneous Solver	Generalized eigenvectors G and eigenvalues σ in the linear system of homogeneous equations $C_0 \times G = \sigma \times A_0 \times G$	M2
	M5: Boundary Conditions	Unknown coefficients a in the system of linear equations, $S \times a = T$, where matrix S and vector T are determined from the boundary conditions (2).	M1, M3, M4
Stage 2	M6: Diffuse Intensity	The diffuse intensity I_d	M3, M4, M5
Stage 3	M7: Reduced Incident Power	The reduced incident intensity power P_{ri}	M10, M12
	M8: Diffuse Power	The diffuse power P_d	M6, M10, M12
Mathematics	M9: Legendre	Associated Legendre Polynomials: $P_l(x)$ and $P_l^m(x)$	
	M10: Specialties	$\log(n!)$ and the Bessel function $J_m(x)$	
	M11: Integration	$\int_a^b f(x)dx = \sum_{i=0}^{n-1} w_i f(x_i)$, where w_i are the weights and x_i are the abscissas	
	M12: Gaussian	Gaussian function $g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu}{2\sigma}}$, where μ is the mean and σ is the standard deviation	
Stores	M13: Data Store	Stores and retrieves the matrices, vectors, numerical results, and other data	
	M14: Power Scattering	Power scattering function $p(\cos\gamma)$], where γ is the scattering angle	M9, M11
	M15: Settings	Establishes specific settings for this problem, defined in table 8	

Table-1 organizes each model into its own functioning layer in addition to summarizing all these models used in the services-oriented design. Moreover, Table-1describes methods of solution as well as the arguments used in all the computations. Stage 1 consists of the linear system equations, which are required to compute the diffuse intensities I_d in Stage 2, as shown in the Services Computing Model in Table-1. The arguments in Stage 1 are k' and ν. Stage 2 consists of a single model M6, which computes the diffuse intensity I_d by using the data from Stage 1. The arguments to Stage 2 are ρ', z', θ, and ψ; see Figure-5 and Table-1.

Afterward, Stage 3 computes the final result: the power received $P = P_d + P_{ri}$, which is the sum of diffuse power and reduced incident power from all the diffuse intensities in Stage 2. The argument to Stage 3 is t'. Stage 3 expects to produce the Power Received. Encapsulating all the mathematical models (Olver *et al.* 2010) on the right side of Figure-2, Table-1 depicts the Data Stores model, providing a repository and retrieval of computational results, such as matrices and vectors, as well as other configurations (e.g., the scattering angle function).

Service Compositions and Orchestrations

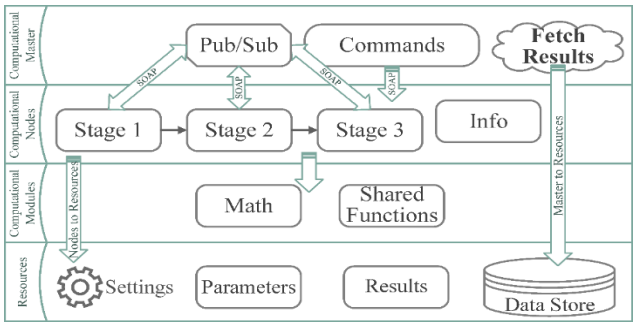

Figure-3. This Service Composition Design Diagram illustrates all the components in the four service layers: (1) The computational-master layer allows service users to visualize numerical results and to initiate a simulation process to a pool of computational nodes. (2) The computational-nodes layer comprises several services hosted in each computational node (i.e., slaves) performing specific tasks. (3) The computational-module layer consists of an auxiliary module that the simulation may use. (4) The resource layer comprises the problem settings as well as the data store for the numerical results and intermediate data.

As shown in the Services Computing Model in Figure-2, the author has categorized all of the 15 computational constituents necessary into various layers to solve the wave propagation problem (designated as various models in Table-1). Each layer functions independently from the others, while only necessitating data output from the preceding stage (i.e., Stage 3 from Stage 2, Stage 2 from Stage 1). The right column in Figure-2 contains multiple Mathematical Models, which can then be requested by the three Stages as necessary. This schematic model allows for the addition of new Mathematical Models without interfering with the infrastructure of the three Stages. As the computational results from each Stage are generated, they can be stored in and retrieved from the Data Stores layer, as illustrated by the left column in Figure-2. The utility of the Services Computing Model is that the user within each layer can alter or redesign the algorithms in that specific layer without affecting the overall architecture or design of other layers.

To empower Services Computing within the scientific computing model in the Service Composition Design Diagram (Figure-3), further encapsulates all layers-including the Computational Master Layer, Computational Nodes Layer (including Stages 1-3 in the Services Computing Model in Figure-2), Computational Modules (which includes the Mathematical Functions in the Services Computing Model), and Resources layer. This overall, broader Service Composition Design seamlessly integrates the Services Computing Model by utilizing a modular design and reassigning each layer within separate categories. The advantage of layering an architecture within another is that each function may reside only within an appropriate layer, while retaining the capability of invoking another function at a different layer through a mutual interface.

When designing architecture for services computing, one must obey the following principles (Huhns and Singh 2005): loosely coupling designs, implementation neutrality, flexible configurability, persistences, and granularity. In a loosely coupled design, the components within a layer (or Stages in the Services Computing Model) are created from an independent framework, so that one does not need to inform the service users that there are changes in the interfaces. Consequently, every service and its inherent functionality shall be independent of one another. For the configurations, one must seek an elegant way to configure the services so that changing the configurations does not affect users in other layers. At any rate, the Services Computing Model and the Service Composition Design have strictly adhered to these guidelines.

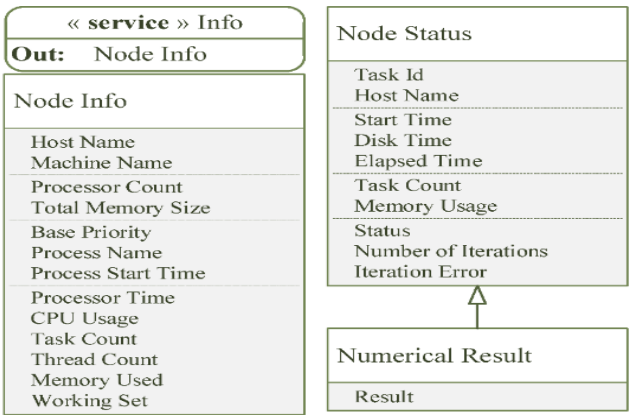

Figure-4. Each computational node provides these two significant structures. The node info provides the necessary information of each computer in the simulator to help facilitate the computational master to schedule a computational task on selected nodes. The node status is the status of a node when the node is performing a task, in which the computational master can take this status to understand a computational progress.

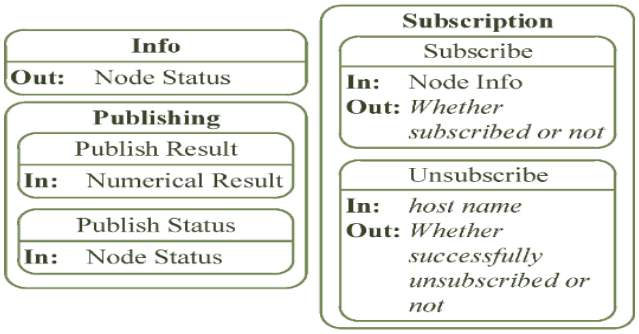

Figure-5. Capable of discovering a computational node, the computational-master layer primarily provides a mechanism for each node to publish data (e.g., statuses and numerical results) and subscribe itself, through the pub/sub services, to the master simulator.

On the whole, the Service Composition Design utilizes master-slave relationships, in which only one master (denoted by “pub/sub" component in the Computational Master Layer in the Service Composition Design in Figure-3) sends commands to many slaves (denoted by Stages 1-3 in the Computational Nodes layer in Figure-3) to perform specific operations. In this paper, referring slaves as computational nodes may help disambiguate the word, slave, because one may relate nodes better as general devices executing specific tasks. In particular, the simulator runs on a master computer within the computational-master layer, and runs the computational nodes on several other computers. The computational master can spearhead commands to each node within the computational-nodes layer through common data structures as shown in Figure-4. Next, subscribed to the master through the pub/sub services, each node then accesses the mathematical functions in the computational-modules layer. Additionally, while

performing a task, each node ought to publish its status and numerical results to the master through the *pub/sub* component in the master layer as shown in Figure-5, thereby parallelizing many calculations at a time.

Operating by the principles of service-oriented architecture (SoA), the simulator utilizing the Service Composition Design can promote seamless communications between every computational node and its computational master by overseeing a scalable but straightforward paradigm. In essence, the simulator has employed custom web services with simple object access protocol (SOAP) because of its conformity with web standards in addition to allowing ease of integrations with various system (MacKenzie *et al*. 2006). Thus, the distributed simulator has facilitated web services to grant computation nodes with appropriate accesses to necessary resources and services.

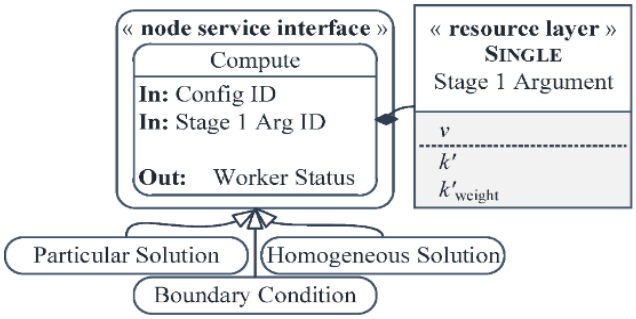

Figure-6. The interface diagram depicts the contracts for stage 1 services (particular solution, homogeneous solution, and boundary condition) with data from the resource layer, including problem configurations identified by Config ID from 1 and a single stage 1 argument (ν and perhaps k′ and k'_{max}) identified by Stage 1 Arg ID.

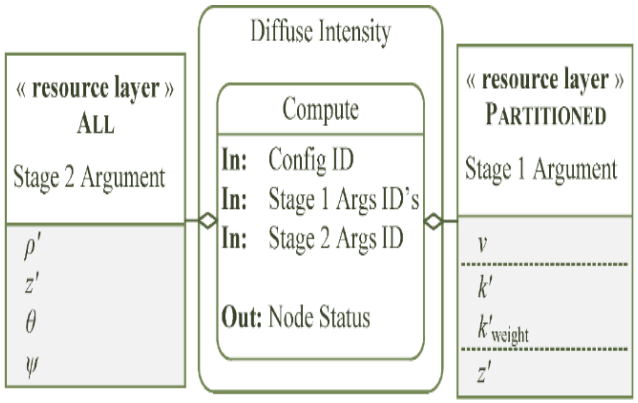

Figure-7. This diagram shows the contract for stage 2 services (diffuse intensity) with data from the resource layer, including problem configurations identified by Config ID from 1 as well as segments of stage 1 arguments (ν and perhaps k′ and k'_{max}) identified by Stage 1 Args ID and segments of stage 2 arguments (ρ′, z′, θ, and ψ) identified by Stage 2 Args ID.

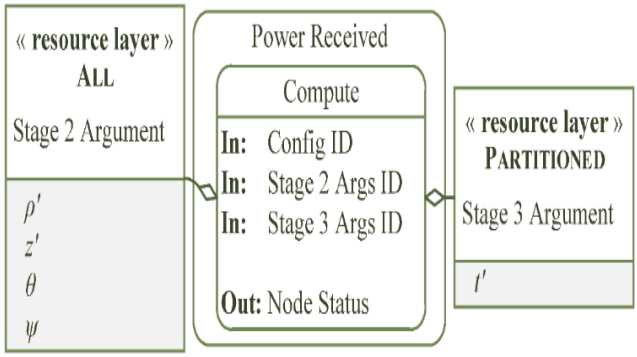

Figure-8. This diagram illustrates the contract for the stage 3 service (power received) with data from the resource layer, including problem configurations identified by Config ID from Figure-1 and segments of stage 2 arguments (ρ′, z′, θ, and ψ) identified by Stage 2 Args ID as well as stage 3 arguments (t′) identified by Stage 3 Args ID. The service immediately responds with a node status while the node is computing in the background.

When a user initiates a simulation by specifying the problem settings in the resources layer, the master service takes these settings and breaks down the calculations into several segments by using a divide-and-conquer scheme. Afterward, the master enqueues these segments of computations to a pool of nodes. Following a specific routine, the computational master subsequently sends specific messages and arguments to the appropriate computational stages-expressed as *Stage 1* in Figure-6, *Stage 2* in Figure-7, and *Stage 3* in Figure-8 within the computational nodes layer, as shown in Figure-3. Each node can then perform these required calculations simultaneously. Afterward, a user sends a request to the master to consolidate all the data and numerical results from the resources layer into the outcome.

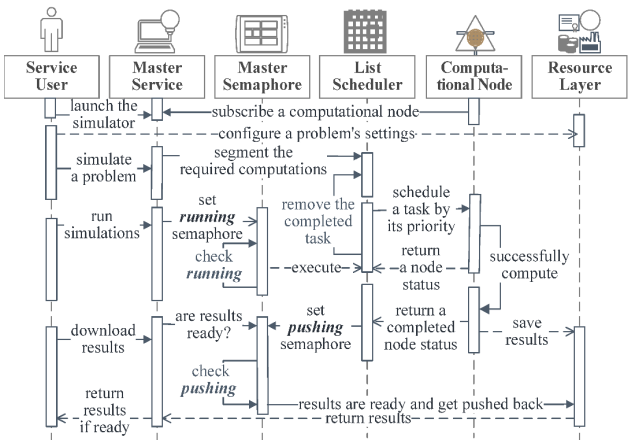

Figure-9. This sequence diagram illustrates how the simulator schedules selected tasks to the nodes.

Demonstrated in Figure-9, the list scheduling scheme used by this simulator allows the computational master to schedule a list of tasks with some priorities to selected nodes (Wang and Sinnen 2018) through a divide-and-conquer scheme, which divides more extensive

calculations into smaller tasks on multiple computational nodes and then conquering later on (e.g., (Yim, Cushman, and Univ. of Nevada Reno 2017)). List scheduling is useful when each computational node knows what to do. Therefore, by identifying all the segments for some determined stage's arguments, the master can partition an extensive calculation into several, smaller, and independent parts so that each computational node can process a scheduled task individually. List priorities, on the other hand, indeed refer to computational dependencies because some calculations may require one another to complete beforehand. For instance, calculating the boundary condition requires both the particular solution and homogeneous solution of the same argument computed beforehand. Another example is that commencing stage 3 requires the completion of both stage 1 and stage 2; refer to how the arrows flow in Figure-2.

In principle, the scheduler partitions the calculations so that each node can receive a specific request of what to calculate. The scheduler then enqueues and assigns these tasks to several nodes depending upon the performance of each node and the number of nodes that are available. After receiving a requesting, the node responds back with its status and information as described in structures in Figure-4. Upon the completion of a task, the node sends a message to the computational master, indicating that the node has completed the work and is ready for a new job; refer to the sequence diagram in Figure-9.

VALIDATIONS AND RESULTS

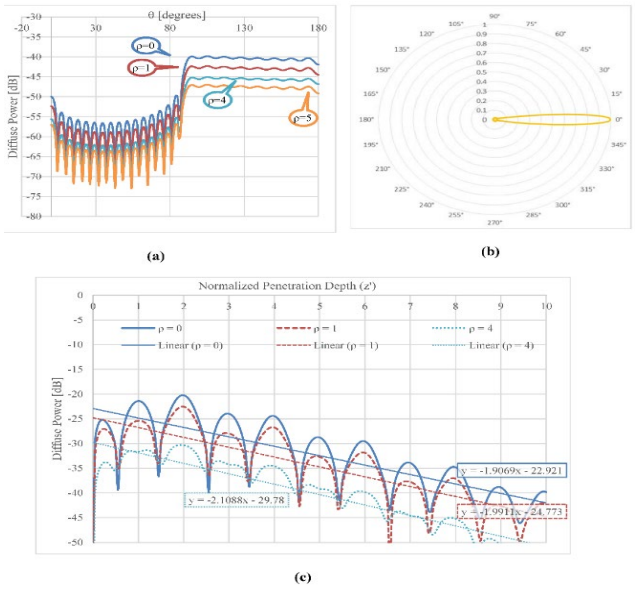

Figure-10. Actual numerical results calculated by the simulator: (a) shows several curves of diffuse power [dB] vs θ for $t' = 0.5$, $z' = 0$, and $\psi = 0$. (b) shows the radiation pattern of a diffuse power normalized to one versus θ when $t' = 0.5$, $z' = 2$, $\rho' = 0$, and $\psi = 0, \pi$. (c) shows the diffuse power in dB plotted against the penetration depth z' when $t' = 0.5$, $\theta = 0$, and $\psi = 0$.

Before benchmarking the simulator, one must ensure whether the simulator is working properly or not through executions of several test cases with known characteristics. First, as illustrated in Figure-10(a), the simulator must satisfy the boundary conditions in (3), in which the diffuse intensity and diffuse power at the vegetation's boundary must be negligibly small. Second, in Figure-10(b), the radiation pattern at the main direction must be consistent with the characteristics of the vegetation model while agreeing with the plots in (Whitman, Schwering, and Wu 2007). Third, as shown in Figure-10(c), the diffuse intensity must exponentially and linearly decay at large penetration depths, $z' \gg 0$. Once validated, the simulator is ready to be benchmarked.

Table-2. A summary of percentages of orchestrations in comparison to the overall simulated times: (orchestration time)/(orchestration time + data time + computational time) for various computational stages with the best efficiencies and the worst efficiencies in distributed simulations.

	Distributed	**Orchestration**
Stage	**Best Ratio**	**Worst Ratio**
1	$2.16\%\left(8\frac{threads}{node}\right)$	$6.05\%\left(1\frac{thread}{node}\right)$
2	$0.0039\%\left(1\frac{thread}{node}\right)$	$0.035\%\left(4\frac{threads}{node}\right)$
3	$0.12\%\left(4\frac{threads}{node}\right)$	$0.78\%\left(1\frac{thread}{node}\right)$
Overall	$0.10\%\left(4\frac{threads}{node}\right)$	$0.57\%\left(1\frac{thread}{node}\right)$

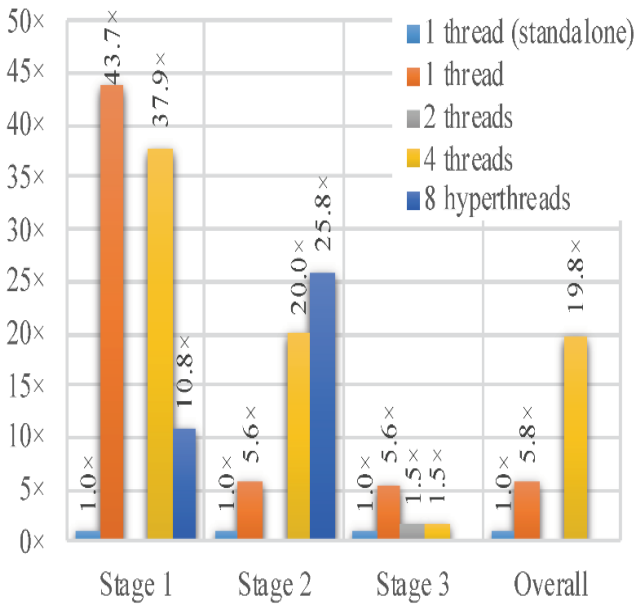

Figure-11. The bar chart shows speedups of 32, 64, 128, and 256 simultaneous tasks all running on 32 computational nodes in contrast to tasks running on a standalone simulator.

Using a massive list of logs from each simulation, one can calculate the efficiencies of service orchestrations

(accesses to services) summarized in Table-2 by comparing service-access times against the overall computational times and data-access times. then concludes with computational speedups of all the stages over a range of runtime configurations. To conclude, in comparison to (Cohen *et al*., n.d.), though a different type of problem, both simulations have exhibited similar speedups of 20 times for 32-node processes with 4 threads per node[1], in which (Cohen *et al*., n.d.) utilizes a traditional-distributed computing procedure.

DISCUSSIONS AND RECOMMENDATIONS

The paper has utilized a Services Computing Model (Figure-2) within a Service Composition Design paradigm (Figure-3) to solve to solve a scientifically complex computational problem. In addition, validating the architectural layers in the Service Composition Design has revealed that these layers have shared common characteristics with the reference model, derived in (Kurniawan *et al.* 2019) through a comprehensive review of several studies using meta-analysis techniques. The objective of the simulator is to calculate the power received in the problem illustrated in Figure-1 on clusters of personal computers (Gray 2008). This research aims to demonstrate the steps of transforming a theoretical framework into a functional simulator, using paradigms from services computing so scientists may design solutions for their scientific computing work. To determine the implementation feasibility of services computing, one shall deploy and evaluate the simulator on several machines. Representing a proof of concept, the simulator, developed in C# and. NET Framework, ought to find a solution to the problem in Figure-1 with an incident pulse trains of collimated-beam (Whitman, Schwering, and Wu 2007).

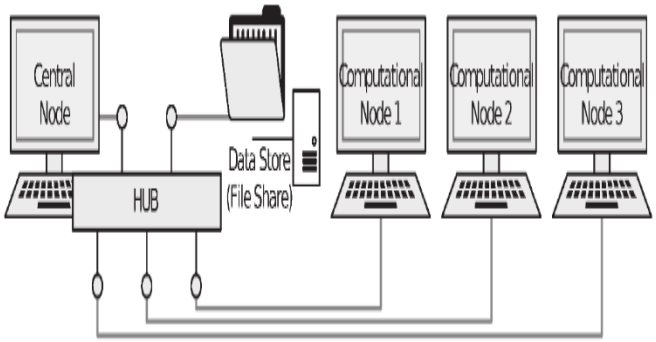

Figure-12. Thirty-three computers, installed with Intel i7-4770 processors and 8 GB of RAM, have their own copies of simulators running.

In conclusion, this article has demonstrated a developmental process of a service-oriented simulator, which solves a complex problem. In particular, the paper makes several recommendations to help scientists make their work more serviceable on the cloud, as follows:

a) Analyze all the formulations and equations required. For instance, the author has taken an understanding of the problem on the propagation and scattering in vegetation and then organized the necessary general functions into three computational stages.
b) Abstract the computational methods into models, as shown in Table-1 author has abstracted each equation as a model via a combination of possible input and output arguments.
c) Refine each model into finer grains iteratively so that other models may reuse these common functionalities. This will enhance each developed Service Composition Design to follow the principle of flexible configurability. Refining models can also help promote heterogeneity by having finer models implement with faster, dedicated processors.
d) Organize the models into architectural layers. For example, this paper has organized all the computational models into five architectural layers, as shown in 2.
e) Identify all Service Interfaces for each Service Computing Model. In practice, one does not need to expose every model as public services. For instance, the author has modeled mutual interfaces between the three computational stages in the Services Computing Model in Figure-2.
f) Outline the algorithms and steps needed to complete the simulations. For example, the sequence diagram in Figure-9 has demonstrated a custom-scheduling system for a better understanding of how the simulator schedules the tasks.
g) Identify all the technologies that best fit the need of the simulator. Examples include programming languages, scheduling system (e.g. message-passing interface (Forum 2015)), frameworks, and hosting solutions (e.g., Amazon EC2 (“Amazon Ec2,” n.d.)). For instance, the author has opted a custom solution using Microsoft C# and .NET Framework. Using the Windows Communication Framework (WCF) from the .NET Framework, the simulator takes advantage of simple-object-access protocol (SOAP) for sending messages to various nodes through a publish/subscribe design pattern in Figure-5.
h) Construct the simulator. The author has developed all the codes in C# with .NET Framework by using Visual Studio as the integrated development environment: each architectural layer is organized into a separate project within a Visual Studio solution.
i) Validate the simulator. One way to validate the codes is to compare the results with well-known characteristics. For instance, the author has shown that the simulator is adequate by observing the three behaviors in Figure-10.
j) Deploy and configure the simulator for the final runs. The author has listed out a few suitable arguments in Table-1 and Table-4 for executing the simulator. Afterward, the author deploys the simulator in Figure-12. The author has leveraged a shared-file system for configuring each simulation and for data transfers.

CONCLUSIONS

This paper proposes an intricately designed and successful simulator that uses services computing principles in solving a complex scientific problem. In doing so, the author is advocating scientists to consider solving their computationally challenging problems using a services computing framework. To promote the transition of scientists over to using services computing, the author has introduced ten main Service Composition Design thought processes that are necessary to implement a loosely coupled and flexibly configurable model for solving computational challenges in scientific fields.

Appendix A: Formulations of the Wave Propagation and Scattering in Vegetation

Studies (Whitman *et al*. 1996; Whitman, Schwering, and Wu 2007; Whitman, Wu, and Schwering 2010) has provided full derivations to the problem illustrated in Figure-1. This section, nonetheless, provides a quick summary of these derivations.

A typical theory used to solve the problem in Figure-1, the Boltzmann transport theory primarily describes the statistical behavior of a random distribution of particles by considering a probability distribution rather than the position and momentum of each particle. Similarly, taking the form of the Boltzmann transport equation, the radiative transfer equation (RTE) also considers a probability distribution of each particle, which scatters and absorbs energies, to determine the changes in the intensity-measured in energy per unit cross-sectional area. Parameters, σ_a and σ_s, signify the absorption and scattering processes of particles in the vegetation while the power scattering function $\mathrm{p}(\cos\gamma)$ describes how a particle scatters and absorbs these energies at an angle γ between the incident power direction $\hat{\mathbf{s}}$ and the outgoing power direction $\hat{\mathbf{s}}'$. Collectively, a combination of these processes is the extinction cross-section, $\sigma_t = \sigma_a + \sigma_s$, forming the albedo of the random media, $W_0 = \frac{\sigma_s}{\sigma_t}$. (Ishimaru 1978 ch. 2).

Governed by the RTE, the specific intensity I is the conventional sum of the reduced-incident intensity I_{ri} and the diffuse intensity I_d. The reduced-incident intensity is the attenuating flux due to scattering and absorption within the vegetation/random media. The diffuse intensity, on the other hand, is the flux created by random media due to multiple scatterings of random objects. Accordingly, the solution expresses the reduced-incident intensity in analytical terms based on a statistical model of the incident wave and the diffuse intensity in terms of integral, partial differential equations in cylindrical and spherical coordinates. That is, the specific intensity I, the reduced-incident intensity I_{ri}, and the diffuse intensity I_d are defined as follows (see Table-3 and Table-4 for descriptions of all the variables):

$$\begin{aligned} I(\mathbf{r}', t', \hat{\mathbf{s}}') &= I_{ri} + I_d, \\ \tfrac{\partial I_{ri}}{\partial t} + \hat{\mathbf{s}} \cdot \nabla I_{ri} + I_{ri} &= 0, \\ \tfrac{\partial I_d}{\partial t} + \hat{\mathbf{s}} \cdot \nabla I_d + I_d &= \tfrac{W_0}{4\pi} \iint_{4\pi} \mathrm{p}\,(\hat{\mathbf{s}} \cdot \hat{\mathbf{s}}') I(\mathbf{r}', t', \hat{\mathbf{s}}') \mathrm{d}\Omega', \end{aligned} \quad (1)$$

where, in cylindrical coordinates of the positions (ρ', z', ϕ) and spherical coordinates of the scatterings (ϕ_s, θ),

Table-3. This table summarizes all the arguments, the names of the arguments, their transform methods, their transformed equivalences, and their maximums. Under each incident wave, the bolded text is the process of transforming from a variable into a discrete variable, indicated by the range below the text.

Category	Name	Argument	Discretization for each Incident Wave		
			Plane-wave	Collimated	Spherical
Periods	Time	t'	Fourier-Cosine Series $\nu \in \{0, 1, \ldots, \nu_{\max}\}$		
Positions	Radial Distance	ρ'	Invariant in the ρ'-direction.	Fourier-Hankel Transform $m \in \{0,1 \ldots, N\}$ and $0 \le k' \le k'_{\max}$	
	Penetration Depth	z'	The incident wave is invariant in the z'-direction.		Fourier-Cosine Transform $-u'_{\max} \le u' \le u'_{\max}$
Scatterings	Azimuth Displacement*	ψ	Spherical Harmonics $m = 0$ and $l \in \{0,1, \ldots, N\}$	Spherical Harmonics $m \in \{0,1, \ldots, N\}$ and $l \in \{m, m+1, \ldots, N\}$	
	Zenith Angle	θ			

*$\psi = \phi - \phi_s$, where ϕ is the positional azimuth, and ϕ_s is the scattering azimuth.

$$\begin{aligned} \hat{\mathbf{s}} \cdot \nabla \equiv \quad & \sin\theta\cos\psi \frac{\partial}{\partial \rho'} - \frac{1}{\rho'}\sin\theta\sin\psi \frac{\partial}{\partial \psi} + \cos\theta \frac{\partial}{\partial z'}, \\ \hat{\mathbf{s}} \cdot \hat{\mathbf{s}}' \equiv \quad & \cos\gamma = \cos\theta\cos\theta' + \sin\theta\sin\theta'\cos(\psi - \psi'), \\ \mathbf{r}' = \quad & \hat{\boldsymbol{\rho}}'\rho + \hat{\mathbf{z}}'z', \ \psi = \phi - \phi_s, \text{ and } \mathrm{d}\Omega' = sin\theta'\mathrm{d}\theta'\mathrm{d}\psi'. \end{aligned} \quad (2)$$

Table-4. This table lists all the problem's configurational settings with their normalized-default values and their descriptions. The vegetation parameters (α, γ_s, W_0, and γ_M) are empirical, taken from one study (Rogers *et al.* 2002), which demonstrates how altering these parameters that characterize a vegetation can affect the attenuation of incident waves within the vegetation.

Param	Value	Description
N	31	Upper bound of the spherical harmonics
P_T	4	Average power, Poynting vector
W_0	0.95	Albedo
σ_t	1	Extinction cross-section per unit area
λ_0	1	Free-space wavelength
T'	2	Period of time
α_0	$4\sqrt{5}$	Inverse pulse width of the incident collimated beam wave pulses
α	0.5	Ratio of the forward scattered power to the total scattered power
$\Delta\gamma_s$	0.105	Front lobe's width of the stage function
$\Delta\gamma_M$	0.021	Front lobe's width of the receiving antenna
w'	1	Beam width of the incident collimated beam wave pulses

In a boundary value problem, to solve the differential equation is to calculate the values of the unknown coefficients in the equation by satisfying the boundary conditions. Henceforth, both reduced-incident and diffuse intensities must satisfy the boundary conditions in the radiance field; that is,

$$\begin{aligned} & \mathrm{I_{ri}} = \mathrm{I_{incident}}, \quad \mathrm{I_d} = 0 \quad \text{at } \mathrm{z'} = 0, \ 0 \le \theta \le \tfrac{\pi}{2} \\ & \text{and} \\ & \mathrm{I_{ri}} \to 0, \quad \mathrm{I_d} \to 0 \quad \text{as } \mathrm{z'} \to \infty. \end{aligned} \quad (3)$$

Applying the Fourier-Hankel transforms and the spherical harmonics to the radiative transfer equation of the diffuse intensity has transformed the partial differential equation into a system of linear, ordinary differential equations. This boundary value problem is normally solved by separating this system of linear equations into a system of particular equations and a system of homogeneous equations together with the boundary conditions. For a summary of the variables and their transform pairs, refer toTable-3. The diffuse intensity becomes

$$I_d = \sum_{m=0}^{\infty} \int_{k'=0}^{\infty} \tilde{A}_m(k')\mathrm{J}_m(k'\rho')\cos(m\psi)k'\mathrm{d}k', \quad (4)$$

where $\mathrm{J}_m(k'\rho')$ is the Bessel function, with a transform of the associated Legendre polynomials $\mathrm{P}_l^m(\cos\theta)$as

$$A_m = \sum_{l=m}^{\infty} A_{ml}(\theta)\mathrm{P}_l^m(\cos\theta), \quad (5)$$

thus forming a complete spherical harmonics. The inverse Fourier-Hankel Transform is defined as follows:

$$A_m = \int_{\psi=0}^{\infty} \int_{\rho'=0}^{\infty} \tilde{I}_d \, \mathrm{J}_m(k'\rho')\cos(m\psi)\rho'\mathrm{d}\rho'\mathrm{d}\psi' \quad (6)$$

After various mathematical manipulations, the coefficients for the diffuse intensity defined in (4)-(6) are

$$\mathrm{A_m} = \begin{cases} \mathrm{F_{ml}} + \sum_{\mathrm{l=m}}^{\infty} \sum_{\mathrm{j}} \mathrm{a_j G_{ml,j}} \mathrm{e}^{-\mathrm{z}/\sigma_\mathrm{j}} & \text{either or} \\ \mathfrak{F}\{\mathrm{F_{ml}}\} + \sum_{\mathrm{l=m}}^{\infty} \sum_{\mathrm{j}} \mathrm{a_j G_{ml,j}} \mathrm{e}^{-\mathrm{z}/\sigma_\mathrm{j}} & \text{if a transform, is needed} \end{cases} \quad (7)$$

where $\mathfrak{F}\{\mathbf{F}_{ml}\}$ is the Fourier-Cosine transform defined as follows:

$$\begin{aligned} \mathrm{F}(u') = \quad & \mathfrak{F}\{\mathrm{f}(z)\} \quad = \frac{2}{\pi}\int_0^{\infty} \mathrm{f}(z)\cos(u'z')\mathrm{d}z' \\ \mathrm{f}(z') = \quad & \mathfrak{F}^{-1}\{\mathrm{F}(u)\} \quad = \int_0^{\infty} \mathrm{F}(u)\cos(u'z')\mathrm{d}u'. \end{aligned} \quad (8)$$

F is the particular solution found from the linear system of equations,

$$\mathrm{B_0} \times \mathrm{F} = \mathrm{f}, \quad (9)$$

where $\mathbf{B_0}$ is the sparse matrix (as illustrated by all the dots in Figure-13) and f is the forcing term formulated by transforming the power scattering function $\mathrm{p}(\cos\gamma)$ from (1) in the same way as (4). G and σ are the eigenvectors and eigenvalues from the homogeneous solution found from the linear system of homogeneous equations,

$$\mathrm{C_0} \times \mathrm{G} = \sigma \times \mathrm{A_0} \times \mathrm{G}, \quad (10)$$

where $\mathrm{C_0}$ and $\mathrm{A_0}$ are sparse matrices (see the caption in 14) and **a** is the unknown coefficients found in the boundary condition (11). Finally, the unknown coefficients a_j in (7) are calculated by solving the linear system of equations,

$$\mathrm{S} \times \mathrm{a} = \mathrm{T}, \quad (11)$$

where matrix S and vector T is a vector are formulated by satisfying all the boundary conditions in (3) and taking considerations of the orthogonal properties of the Bessel function $\mathrm{J}_m(x)$ and associated Legendre polynomials $\mathrm{P}_l^m(x)$.

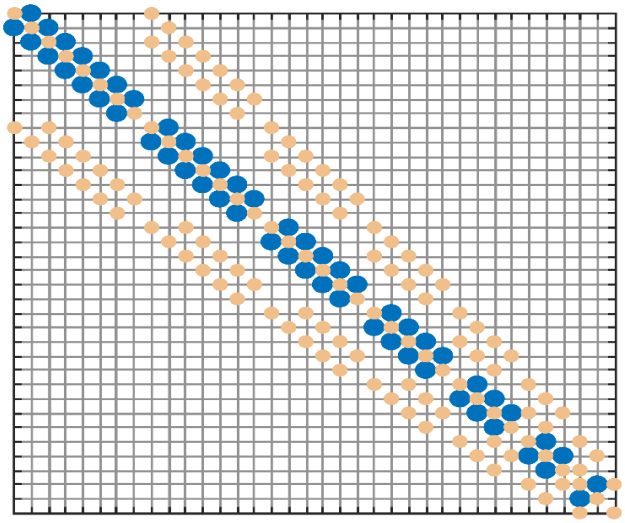

Figure-13. This spy chart shows, as solid circles, all the non-zero values of the sparse matrix B_0 from (9) for N = 7. Likewise, all the lighter and smaller dots represent non-zero values of the sparse matrix $\mathbf{C_0}$ from (10) using the same value of N, whereas all other circles are for the matrix A_0.

The power received can directly be expressed, in terms of an expansion of Fourier-series, as

$$P(t';\rho',z';\theta',\psi') = P_{ri} + P_d \propto \{\sum_{v=0}^{\infty}[I_{ri} + I_d]\, e^{iv\omega'(t'-z')}\} \quad (12)$$

with the diffuse intensities (I_d) defined in(4)-(11) and the reduced-incident intensity and power (I_{ri} and P_{ri}) derived analytically from (1)based on the characteristics of the incident beam wave.

REFERENCES

AbdelBaky M., M. Parashar Hyunjoo Kim, K. E. Jordan, V. Sachdeva, J. Sexton, H. Jamjoom, *et al.* 2012. Enabling High-Performance Computing as a Service. Computer. 45(10): 72-80.
Amazon Ec2. n.d. https://aws.amazon.com/ec2/.

Bichier M. and K.-. Lin. 2006. Service-Oriented Computing. Computer. 39(3): 99-101.

Bouguettaya Athman, Munindar Singh, Michael Huhns, Quan Z Sheng, Hai Dong, Qi Yu, Azadeh Ghari Neiat, *et al*. 2017. A Service Computing Manifesto: The Next 10 Years. Communications of the ACM. 60(4): 64-72.

Chen Jianhui, Jianhua Ma, Ning Zhong, Yiyu Yao, Jiming Liu, Runhe Huang, Wenbin Li, Zhisheng Huang, Yang Gao and Jianping Cao. 2014. WaaS: Wisdom as a Service. IEEE Intelligent Systems. 29(6): 40-47.

Cohen Jeremy, Ioannis Filippis, Mark Woodbridge, Daniela Bauer, Neil C. Hong, Mike Jackson, Sarah Butcher, et al. 1983. n.d. RAPPORT: Running Scientific High-Performance Computing Applications on the Cloud. Philosophical Transactions: Mathematical, Physical and Engineering Sciences. 371: 1-15.

Forum Message Passing Interface. 2015. MPI: A Message-Passing Interface Standard Version 3.1. In, 1-8. https://www.mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf; University of Tennessee, Knoxville.

Gray Jim. 2008. Distributed Computing Economics. Queue. 6(3): 63-68.

Huhns Michael N and Munindar P Singh. 2005. Service-Oriented Computing: Key Concepts and Principles. IEEE Internet Computing. 9(1): 75-81.

Ishimaru Akira. 1978. Wave Propagation and Scattering in Random Media. Vol. 2. Academic press New York.

Keyes David E, Lois C McInnes, Carol Woodward, William Gropp, Eric Myra, Michael Pernice, John Bell, et al. 2013. Multiphysics Simulations: Challenges and Opportunities. The International Journal of High Performance Computing Applications. 27(1): 4-83.

Kurniawan Novianto B., Suhardi, Arry A. Arman, Yoanes Bandung and Purnomo Yustianto. 2019. A Reference Model of Services Computing Systems Platform Based on Meta-Analysis Technique. Service Oriented Computing and Applications. 13(1): 31-49.

MacKenzie C Matthew, Ken Laskey, Francis McCabe, Peter F Brown, Rebekah Metz and Booz Allen Hamilton. 2006. Reference Model for Service Oriented Architecture 1.0. OASIS Standard. 12: 18.

Olver Frank WJ, Daniel W Lozier, Ronald F Boisvert and Charles W Clark. 2010. NIST Handbook of Mathematical Functions Hardback and Cd-Rom. Cambridge university press.

Papazoglou Michael P, Paolo Traverso, Schahram Dustdar and Frank Leymann. 2007. Service-Oriented Computing: State of the Art and Research Challenges. Computer. 40(11).

Rodriguez Maria Alejandra and Rajkumar Buyya. 2014. Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds. IEEE Transactions on Cloud Computing. 2(2): 222-35.

Rogers Neil C, A Seville, J Richter, D Ndzi, N Savage, R Caldeirinha, AK Shukla, *et al*. 2002. A Generic Model of 1-60 Ghz Radio Propagation Through Vegetation-Final Report. Radio Agency, UK.

Shainer Gilad, Todd Wilde, Pak Lui, Tong Liu, Michael Kagan, Mike Dubman, Yiftah Shahar, Richard Graham, Pavel Shamis, and Steve Poole. n.d. The Co-Design Architecture for Exascale Systems, a Novel Approach for

Scalable Designs. Computer Science - Research and Development. 28(2-3): 119-125.

Simmendinger Christian, Roman Iakymchuk, Luis Cebamanos, Dana Akhmetova, Valeria Bartsch, TiberiuRotaru, Mirko Rahn, Erwin Laure and Stefano Markidis. 2019. Interoperability Strategies for Gaspi and Mpi in Large-Scale Scientific Applications. The International Journal of High Performance Computing Applications. 33(3): 554-68.

Slotnick Jeffrey, AbdollahKhodadoust, Juan Alonso, David Darmofal, William Gropp, Elizabeth Lurie and Dimitri Mavriplis. 2014. CFD Vision 2030 Study: A Path to Revolutionary Computational Aero sciences.

Storer Tim. 2017. Bridging the Chasm: A Survey of Software Engineering Practice in Scientific Programming. ACM Computing Surveys (CSUR). 50(4): 47.

Wang Huijun, and Oliver Sinnen. 2018. List-Scheduling Versus Cluster-Scheduling. IEEE Transactions on Parallel and Distributed Systems. 29(8): 1736-49.

Whitman Gerald M, Felix K Schwering and Michael Y-C Wu. 2007. Collimated Beam Wave Pulse Propagation and Scattering in Vegetation Using Scalar Transport Theory. IEEE Transactions on Antennas and Propagation. 55(6): 1599-1612.

Whitman Gerald M, Michael YC Wu and Felix K Schwering. 2010. Propagation and Scattering of Spherical Wave Pulses in Vegetation Using Scalar Transport Theory. IEEE Transactions on Antennas and Propagation. 58(5): 1662-76.

Whitman G. M., F. Schwering, A. Triolo and N. Y. Cho. 1996. A Transport Theory of Pulse Propagation in a Strongly Forward Scattering Random Medium. IEEE Transactions on Antennas and Propagation. 44(1): 118-28.

Wu Michael Y-C, Jian Lin and Nanfei Sun. 2017. A Simulation Model for Wave Propagation and Scattering of Beam Waves in Vegetation Using Scalar Transport Theory. International Journal of Modelling and Simulation. 37(1): 1-14.

Yim WC, JC Cushman and Reno NV (United States) Univ. of Nevada Reno. 2017. Divide and Conquer (Dc) Blast: Fast and Easy Blast Execution Within Hpc Environments. PEERJ 5: e3486-e3486.