Dynamical systems analysis using many-task interactive cloud computing

Stefan N. Popov¹, Sergei V. Vostokin¹, Anton V. Doroshin¹²

¹ Samara National Research University, Samara, Russia
² The work is supported by the Russian Science Foundation (# 19-19-00085)

stef4n.popov@gmail.com

Abstract. The article states an approach for developing a distributed application architecture for dynamic systems analysis in a hybrid cloud. The problem that was posed as an example is based on the calculation of the Lyapunov characteristic exponents for trajectories of dynamical systems in phase space. Parallelization of the computational process is achieved by supplying the list of running tasks for computing Lyapunov characteristic exponents, depending on the result of finished tasks during calculations. We have implemented a distributed application to make this experiment automated. The architecture of application elements is established on the principles of the enterprise desktop grid and cloud systems. A vital detail of the application architecture is the deployment process based on a cloud hybrid computing environment using the Jupyter server running on the DigitalOcean cloud platform and the Everest cloud platform simultaneously with the Maple package.

1. Introduction
At present, it is becoming more and more interesting to study the processes of nonlinear dynamics and chaotic behavior of complex systems based on numerical models. From a practical point of view, these studies are the basis for the design of different motion control systems, for example, orientation and control systems for spacecraft. From a theoretical point of view, there is interest in studying the mathematical properties of differential equations. With the help of numerical models, it is possible to build maps of phase states for a given system of differential equations. Moreover, there are no alternatives in terms of study tools in the subject domain under consideration except numerical or combined numerical-analytical methods. The main reason is the difficulty or impossibility of obtaining analytical solutions for most practical problems.

On another hand, there are some technical issues that cause it much tricky to apply numerical methods of study for the scope of dynamical systems modeling.

First, the computational complexity of an experiment is high. Getting only one solution with sufficient accuracy takes a lot of time, even on a modern high-performance computer. Although for a detailed analysis it is important to obtain a huge number of solutions with different initial conditions and numerical method settings.

Second, there must be a combination of numerical and systematic instrumental methods. This combination is significant considering that the manual programming of a numerical method is time-
Third, methods for researching phase spaces based on strict search algorithms and strict criteria for catching the satisfying behavior of the dynamical system have rare usage. As an alternative, the search is based on the intuition and expertise of the researcher. This involves the usage of the subjective analysis of visual modeling artifacts. The manual phase diagram analysis can be mentioned as an example. This influences the reproducibility of the results and the completeness of the research.

The first issue does not provide research of the dynamical systems on one machine: high-performance computing technologies should be applied. The second issue produces challenges when involving traditional technologies of high-performance computing on clusters, like MPI and OpenMP. The third issue needs the development of suitable technology for the control of computational experiments.

The paper suggests a solution for these issues, which is based on a compound of verified technologies for organizing enterprise-level desktop grid systems [1] and modern web-based computer modeling systems [2]. The last-mentioned systems improved quickly because of the popularity of data processing, machine learning, and artificial intelligence. We also investigate the usage of hybrid academic and cloud systems [3] for the deployment of our software package.

2. Implementation of the Lyapunov Characteristic Exponent Method

In this section, we shortly state the method for computation of Lyapunov characteristic exponents for the Lorenz system [4] and present the Maple implementation of well-known Benettin’s algorithm with Gram-Schmidt orthogonalization [5].

The calculation of Lyapunov characteristic exponents is based on the numerical integration of the differential equations. In the case of the Lorenz system, we have the following equations:

\[
\begin{align*}
\dot{x} &= \sigma(y - x); \\
\dot{y} &= rx - y - xz; \\
\dot{z} &= -bz + xy
\end{align*}
\]

To get the Lyapunov characteristic exponents we need to integrate the system (1) and the supplemental systems for variations, specifically the three subsystems for variations \(\{\dot{x}_i, \dot{y}_i, \dot{z}_i\}, i = 1, 2, 3:\)

\[
\begin{align*}
\dot{x}_1 &= \sigma(y_1 - x_1); \\
\dot{y}_1 &= r x_1 - y_1 - x_1 z - x z_1; \\
\dot{z}_1 &= -b z_1 + x_1 y + x y_1; \\
\dot{x}_2 &= \sigma(y_2 - x_2); \\
\dot{y}_2 &= r x_2 - y_2 - x_2 z - x z_2; \\
\dot{z}_2 &= -b z_2 + x_2 y + x y_2; \\
\dot{x}_3 &= \sigma(y_3 - x_3); \\
\dot{y}_3 &= r x_3 - y_3 - x_3 z - x z_3; \\
\dot{z}_3 &= -b z_3 + x_3 y + x y_3.
\end{align*}
\]

The integration procedure is performed for all twelve equations (1-2). At each integration step (at time step \(T\)), normalization and orthogonalization of the variation vectors are implemented. The partial sum

\[
S_i = \sum_{k=1}^{N} \ln ||(x_i(T), y_i(T), z_i(T))||
\]

is calculated during the integration procedure. At the end of the procedure, the sum \(S_i\) will be equal to Lyapunov characteristic exponents.

The Maple source code for the integration procedure is shown below. In this code, the Benettin’s algorithm was implemented for the Lorenz system with provided values of \(b, r, \) and \(\sigma\).

```maple
with(plottools):
with(DEtools):
with(plottools,line):with(linalg):

b:=8/3: r:=28: sigma_:=10:

The Lorenz system is shown in the system of equation ur1, ur2, ur3.
```
url1:=diff(x(t),t)=-sigma_*(x(t)-y(t)):
ur2:=diff(y(t),t)=(r*x(t)-y(t)-x(t)*z(t)):
ur3:=diff(z(t),t)=(x(t)*y(t)-b*z(t)):

Three systems in variations:

ur11:=diff(x1(t),t)=-sigma_*(x1(t)-y1(t)):
ur21:=diff(y1(t),t)=(r*x1(t)-y1(t)-x(t)*z1(t)-x1(t)*z(t)):
ur31:=diff(z1(t),t)=(x1(t)*y(t)+x(t)*y1(t)-b*z1(t)):
ur12:=diff(x2(t),t)=-sigma_*(x2(t)-y2(t)):
ur22:=diff(y2(t),t)=(r*x2(t)-y2(t)-x(t)*z2(t)-x2(t)*z(t)):
ur32:=diff(z2(t),t)=(x2(t)*y(t)+x(t)*y2(t)-b*z2(t)):
ur13:=diff(x3(t),t)=-sigma_*(x3(t)-y3(t)):
ur23:=diff(y3(t),t)=(r*x3(t)-y3(t)-x(t)*z3(t)-x3(t)*z(t)):
ur33:=diff(z3(t),t)=(x3(t)*y(t)+x(t)*y3(t)-b*z3(t)):

Initial conditions for phase trajectory representing co-called strange attractor.

x0:=0.5:y0:=1:z0:=1.5:
T:=0.05:Num_of_steps:=10000:LAP1:=vector(Num_of_steps,[]):LAP2:=vector(Num_of_steps,[]):LAP3:=vector(Num_of_steps,[]):
epsilon_resh:=1:

Initial conditions for systems in variations:
nach_01:=vector(3,[epsilon_resh,0,0]):
nach_02:=vector(3,[0,epsilon_resh,0]):
nach_03:=vector(3,[0,0,epsilon_resh]):
S1:=0:S2:=0:S3:=0:

Here is the beginning of the integration procedure.

for i from 1 by 1 to Num_of_steps do:
Resh1:=dsolve( [ur1,ur2,ur3,ur11,ur21,ur31,ur12,ur22,ur32,ur13,ur23,ur33,x(0)=x0,y(0)=y0,z(0)=z0,x1(0)=nach_01[1],y1(0)=nach_01[2],z1(0)=nach_01[3],x2(0)=nach_02[1],y2(0)=nach_02[2],z2(0)=nach_02[3],x3(0)=nach_03[1],y3(0)=nach_03[2],z3(0)=nach_03[3],[x(t),y(t),z(t),x1(t),y1(t),z1(t),x2(t),y2(t),z2(t),x3(t),y3(t),z3(t)], type=numeric, method=rkf45,relerr=0.000001,abser=0.000001, output=listprocedure):
X1 := eval(x1(t),Resh1):
Y1 := eval(y1(t),Resh1):
Z1 := eval(z1(t),Resh1):
X2 := eval(x2(t),Resh1):
Y2 := eval(y2(t),Resh1):
Z2 := eval(z2(t),Resh1):
X3 := eval(x3(t),Resh1):
Y3 := eval(y3(t),Resh1):

The numerical solution based on the integration procedure is called Resh1:

X := eval(x(t),Resh1):
Y := eval(y(t),Resh1):
Z := eval(z(t),Resh1):
xyzT:=vector(3,[X(T),Y(T),Z(T)]):
xyz1T:=vector(3,[X1(T),Y1(T),Z1(T)]):
xyz2T := vector(3, [X2(T), Y2(T), Z2(T)]):
xyz3T := vector(3, [X3(T), Y3(T), Z3(T)]):

Doing normalization, orthogonalization and calculating of the partial sums $S_i$:

\[
\text{norm}_{xyz1T} := \text{norm}(xyz1T):
S1 := S1 + \ln(\text{norm}_{xyz1T}/\epsilon_{resh}):
xyz1_0 := \text{evalm}(xyz1T/norm_{xyz1T}):
xyz2_1 := xyz2T - \text{evalm}(\text{multiply}(xyz2T, xyz1_0)*xyz1_0):
xyz2_0 := \text{evalm}(xyz2_1/norm(xyz2_1)):
S2 := S2 + \ln(\text{norm}(xyz2_1)/\epsilon_{resh}):
xyz3_1 := xyz3T - \text{evalm}(\text{multiply}(xyz3T, xyz1_0)*xyz1_0) - \text{evalm}(\text{multiply}(xyz3T, xyz2_0)*xyz2_0):
xyz3_0 := \text{evalm}(xyz3_1/norm(xyz3_1)):
S3 := S3 + \ln(\text{norm}(xyz3_1)/\epsilon_{resh}):
\]

We get new initial conditions for systems on the next iteration:

nach_01 := xyz1_0:nach_02 := xyz2_0:nach_03 := xyz3_0:
x0 := X(T): y0 := Y(T): z0 := Z(T):

Fixing the current values of Lyapunov characteristic exponents and finishing the iteration.

LAP1[i] := S1/(i*T): LAP2[i] := S2/(i*T): LAP3[i] := S3/(i*T):
end do:

The triple of real numbers \{LAP1[\text{Num_of_steps}]; LAP2[\text{Num_of_steps}]; LAP3[\text{Num_of_steps}]\} is the output of the calculation procedure. The test runs of the Maple worksheet gave the \{0.88; 0; -14.4\} triple of the Lyapunov characteristic exponents for the initial conditions presented in the Maple code above, which is in good agreement with the calculations known from the literature. The typical calculation of the one triple on Intel Core i7-4510u 2.6 GHz computer (4 Gb, 64-Windows10) with Maple 10 installed took approximately 270 seconds (4.5 minutes). It means that carrying out a series of experiments requires parallelization. In the next section, we will discuss a software architecture for running parallel experiments with varying initial conditions.

3. Architecture of the Software Package

When performing parallel calculations, the researcher is interested in the ability to work with the source code in Maple language without rewriting it into other programming languages. He also needs the ability to interact with the program, in the same way as in the Maple environment. Simple configuration of multiple computing resources is preferred. It is also important to be able to collaborate with other researchers: jointly manage the experiment and use each other’s computing resources with the Maple package installed. Let’s consider the software architecture and the deployment of its components that meet the specified requirements (Figure 1).

In Figure 1, we can see that the software system consists of three components: (a) a management component, (b) an intermediary component, and (c) a computing component.
Figure 1. Software architecture for conducting parallel experiments on a dynamical system.

The management component is used to generate parallel tasks with different initial conditions for the procedure discussed in Section 2. The management component consists of two parts: the first part is a Jupyter notebook that generates the parallel tasks; the second part is a library that allows users to interact with the Everest platform server [6] using the REST protocol. Both components of the system are implemented using the C++ programming language. The libcurl library is used for creating networks. Jupyter notebook uses the Xeus-cling core to execute the C++ source code. One of the main features of our implementation of task parallelism [7] is the ability to dynamically generate new tasks based on the results of completed tasks.

The management component is deployed in conjunction with the JupyterHub [8] server in the DigitalOcean [9] cloud service. The component source code is stored in the GitHub service repository, and then it is deployed on the DigitalOcean virtual machine using nbgitpuller. After ending the automated deployment operation, the researcher works interactively with the management component through a web browser.

An intermediary component is a special program that is pre-installed on the Everest server. The installation process is executed once for a number of experiments using the Everest graphical user interface (web GUI) provided by the Everest platform.

The Everest application determines how to handle task parameters that are automatically passed from the management component to the Everest server to call the task on the computer component. The Everest application also states a method for transferring the task result from the computer component to the management component. The Maple spreadsheet that calculates Lyapunov exponents is uploaded to the Everest server as part of the program.

The Everest resource agent represents a computing component. We created an installer to configure and maintain the agent on the computing resource. After the installation is complete, the agent is registered on the Everest server. Both regular workstations and virtual machines with the pre-installed Maple package can be used as computing resources. We use the VMware virtual desktop service (VMware Horizon web client) in the corporate cloud of Samara University. The service was chosen because it allows you to run virtual machines from a web browser that simplify configuration.

In this way, the software for studying dynamical systems is launched and configured entirely through a web browser. The researcher works with the software package in the familiar paradigm of the REPL (read-eval-print loop) interface, supported by both Maple and Jupyter.
4. Discussion of Computing Experiment

In our earlier research we measured the standard execution time for calculating Lyapunov exponents; evaluated how the overhead of task management affects the efficiency of calculations; studied the possibility of performing a series of experiments without failures and the correct interaction between components in the proposed architecture [11, 12].

In this article, we implement an experimental workflow for investigating the properties of the Lorenz equation to show the real acceleration of cloud computing using up to 10 virtual machines. In the workflow, we try to maximize the $L_1$ component of the Lyapunov spectrum by varying the values of $b$, $r$, and $\sigma$ using simple stochastic optimization.

We run 12 series of the optimization: 3 series on 1 virtual machine, 3 series on 2 virtual machines, 3 series on 5 virtual machines, and 3 series on 10 virtual machines. Each series consists of 10 independent tasks with different values of $b$, $r$, and $\sigma$. We measured execution time for each series and calculate average speedup ($S$) and efficiency ($E$) for the three runs with the same number of virtual machines. The experimental results of parallel Lyapunov exponents calculation are shown in Table 1.

| p, VM number | $T_p$, s | $\bar{T}_p$, s | $S = \frac{\bar{T}_1}{\bar{T}_p}$ | $E = \frac{S}{p}$ |
|--------------|----------|----------------|-------------------------------|-----------------|
| 1            | 2581.97  | 2552.28        | 1.000                         | 1.000           |
| 1            | 2588.17  | 1292.56        | 1.974                         | 0.987           |
| 2            | 2486.69  | 546.384        | 4.671                         | 0.934           |
| 5            | 556.235  | 277.212        | 9.207                         | 0.921           |
| 10           | 274.309  | 277.212        | 9.207                         | 0.921           |

The table shows that in computational experiments we managed to achieve high efficiency of more than 90 percent in the worst case (on 10 virtual machines), as predicted in preliminary load testing [12]. Analyzing the absolute complexity of the calculations, we can conclude that the time saving when performing all 12 series of experiments on 10 virtual machines is 7 hours 35 minutes. The total computation time in the case of sequential calculations is $12 \times T_1 \sim 12 \times 2552.28 = 30 627.32$ seconds $\sim 8$ hours 30 minutes. The total computation time when performing parallel computations on 10 virtual machines is $12 \times T_{10} = 12 \times 277.212 \sim 55$ minutes, which gives a significant gain in time of almost 7 and a half hours. Starting with $r = 28.0$, $s = 10.0$, $b = 2.66666$ and $L_1 = 0.88$, we get $r = 33.1677$, $s = 12.27$, $b = 4.33019$, and $L_1 = 1.16316$, thus found a more chaotic behavior.

5. Conclusions

In this article, we have proposed and tested an architecture for investigating the behavior of dynamical systems using many-task interactive cloud computing. We have also developed a software package that allows users to automate resource-intensive computational experiments using this method. The novelty of the method, which provides an advantage over other, more traditional approaches, is the ability to use together the enterprise desktop grid computing and modern approaches to computing, including web systems and cloud web services.

We tested the technology in the corporate cloud service of Samara University. Our experiments have shown that up to 10 virtual machines can be used together in a single computation with an efficiency of
about 90 percent. Solving a real optimization problem from the field of dynamical systems analysis, we have shown the applicability and usability of the proposed architecture.

Our future plans include using this architecture in conjunction with globally distributed storage. We plan to focus on combining distributed computing and distributed storage and finding the best performing solution for that model.

References

[1] Ivashko, E., Chernov, I., Nikitina, N.: A survey of desktop grid scheduling. IEEE Transactions on Parallel and Distributed Systems, vol. 29, No. 12, pp. 2882-2895. IEEE (2018).

[2] Bernadette M. Randles, Irene V. Pasquetto, Milena S. Golshan, Christine L. Borman: Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study. IEEE Joint Conference on Digital Libraries (JCDL), Toronto, ON, Canada (2017).

[3] Goyal, S.: Public vs private vs hybrid vs community-cloud computing: a critical review. International Journal of Computer Network and Information Security, vol. 6, No. 3, p. 20. MECS Press (2014).

[4] Lorenz, E. N.: Deterministic nonperiodic flow. Journal of the Atmospheric Sciences, vol. 20, pp. 130–141. AMS (1963)

[5] Benettin, G., Galgani, L., Giorgilli, A., Strelcyn, J.-M.: Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems: A method for computing all of them. P.I: Theory. P.II: Numerical application. Meccanica, vol. 15, pp. 9-30. Springer (1980).

[6] Sukhoroslov, O., Volkov, S., Afanasiev, A. A.: Web-Based Platform for Publication and Distributed Execution of Computing Applications. 14th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 175-184. IEEE (2015).

[7] Thoman, P., et al.: A taxonomy of task-based parallel programming technologies for high-performance computing. The Journal of Supercomputing, vol. 74, No. 4, pp. 1422-1434. Springer (2018).

[8] Andrea Zonca, Robert S. Sinkovits: Deploying Jupyter Notebooks at scale on XSEDE resources for Science Gateways and workshops. PEARC ’18, pp. 1-2, Pittsburgh, PA, USA (2018).

[9] Xiaomin Wang, Xin Zhang, Tianhe Chi: Data Processing and Application of Digital Ocean. 2008 Congress on Image and Signal Processing, Sanya, Hainan, China (2008).

[10] Senger, H., da Silva, F. A. B.: Bounds on the scalability of bag-of-tasks applications running on master-slave platforms. Parallel Processing Letters, vol. 22., No. 02, p. 1250004. World Scientific (2012).

[11] Vostokin, S.V., Sukhoroslov, O.V., Bobyleva, I.V., Popov, S.N.: Implementing computations with dynamic task dependencies in the desktop grid environment using Everest and Templet Web. CEUR Workshop Proceedings, Volume 2267, pp. 271-275. CEUR-WS, Dubna (2018).

[12] S.V. Vostokin , A.V. Doroshin: A Software Package for Studying Dynamical Systems Using the Lyapunov Characteristic Exponent in a Hybrid Computing Environment. Russian Supercomputing Days 2020 (2020).