Progress in NUCLEAR SCIENCE and TECHNOLOGY, Vol. 2, pp.591-597 (2011)

ARTICLE

Fault-Tolerant Mechanism of Both Job Execution and File Transfer for Integrated Nuclear Energy Simulation

Takayuki TATEKAWA*, Naoya TESHIMA, Yoshio SUZUKI and Hiroshi TAKEMIYA

Center for Computational Science and e-Systems, Japan Atomic Energy Agency, Tokyo 110-0015, Japan

The integration of several codes to simulate physical processes or components of a nuclear energy facility facilitates large-scale, detailed simulation. However, integrated simulations require several weeks or months of CPU time. We developed a fault-tolerant method for cooperative execution of codes, which avoids unscheduled outage of computers or networks. The method deals with abnormal job terminations on supercomputers and file transfer errors. If a computer causes an unexpected outage, the method attempts to submit the simulation task to an alternative computer. The method also detects transfer errors by comparing the size of files before and after transfer. The relationship between jobs and file transfers is connected by the fault-tolerant method, which allows us to decide the execution order of codes by definition of file flow. This enables the operation on integrated simulations where codes are executed sequentially or concurrently.

KEYWORDS: grid, simulation, fault-tolerance, API, AEGIS

I. Introduction

Large-scale integrated simulations have been developed in the nuclear energy field to investigate physical and engineering behavior at the microscopic and macroscopic level, both in the short-term and long-term. These simulations cannot be executed using a single job class on a computer, so integrated simulations are required that couple several simulation codes, each of which handles a physical process or an engineering component of the overall system.

We developed a full-scale three-dimensional vibration simulator for an entire nuclear power plant (NPP) as a demonstration of a large-scale integrated simulations in the nuclear energy field.[1)] We used the simulation to Predicting Quake-Proof Capability of Nuclear Power Plant, and to investigate a burning plasma simulation.[2)] The full-scale three-dimensional vibration simulation of an entire nuclear power plant, where cooperation of the code allowed analysis of each component's behavior, meant we could realize physical phenomena, such as local deformation caused by interactions between components. The Quake-Proof Capability of Nuclear Power Plant was predicted using the cooperation of codes for each physical phenomenon, including seismic simulation, structure analysis, thermal hydraulics, and nuclear reaction. We predicted the behavior of burning plasma in a tokamak reactor using a burning plasma simulation, where the codes cooperated to deal with current transport, stability analysis, and current drive. [3)] The simulation handled short-term phenomena within microseconds and analyzed the behavior in several minutes. These simulations can have very wide temporal and spatial ranges.

The cooperative integration of simulation code on distributed computers is problematic. Grid computing technology has been used for coupling heterogeneous computers. Workflow tools, such as Kepler[4)] and TME,[5)] enable the construction and execution of integrated simulations on the grid. These tools are easily used by researchers, but applications of these tools on integrated simulations are limited. However, remote procedure calls (RPC)[6)] or grid-enabled MPI, such as STAMPI,[7)] MPICH-G,[8)] or PACX MPI,[9)] can be applied to different integrated simulations, although substantial modifications of codes are required. Therefore, it is necessary to develop a grid computing technology that enables researchers in nuclear energy field to more easily operate a variety of integrated simulations. Furthermore, it is important that substantial modifications of each code are not required by any new technology.

Cooperative execution of codes requires file flow between the codes. We developed Simple Orchestration Application Framework (SOAF)[2,3)] where the cooperative execution of codes is based on file flow. In this framework, users run integrated simulations on remote supercomputers from a client PC. No substantial modification of simulation codes is required. SOAF can handle a variety of integrated simulations.

It is necessary to reduce the amount of effort researchers spend on client application development. SOAF is implemented using application program interfaces (APIs) on the grid infrastructure of the Atomic Energy Grid InfraStructure (AEGIS).[1,10-12)] Users can easily develop a grid-enabled client application using AEGIS APIs.

SOAF uses a fault-tolerance (FT) method[13)] when conducting long-term integrated simulations. Integrated simulations can require several weeks or months of CPU time. When a job is stopped by an execution time limit or an

*Corresponding author, E-mail: tatekawa.takayuki@jaea.go.jp unexpected outage of computer, the method automatically resubmits the job to same or an alternative computer.

We also need to handle file transfers in long-term integrated simulations. The files transferred between codes can be huge in large-scale simulations, which means that file transfers can take a long time. It is important to correctly handle file transfers during computer or network outages. Furthermore, when a job is submitted to alternative computers by the FT mechanism, the file transfer processes can change. SOAF cannot adequately deal with file transfer problems.

We developed a FT method for job execution and file transfer in integrated simulations performed on AEGIS. The FT method is also implemented on the AEGIS API (referred to as the "FT API"). When an unscheduled computer outage occurs, the FT method resubmits a terminated job. After resubmission, the computer and its directory corresponding to file transfer destinations are automatically modified. The FT method also manages file outputs on departure and transfers them to their destinations immediately after file output is completed. The completion of file transfer is confirmed by comparing file sizes before and after transfer. When a file transfer fails, the FT method automatically retries file transfer.

During FT API development, we investigated file flow between simulation codes. We defined the relationship between simulation codes and files transferred between codes, which made it easier to handle the integrated simulation from a client PC.

II. Fault-Tolerant Mechanism of Job Execution

1. AEGIS Client API

AEGIS is grid middleware that we developed for atomic energy research. The schematic diagram of AEGIS is shown in **Fig. 1**. Users operate jobs on supercomputers from a client PC in AEGIS. Communication between the client PC and the supercomputer is secure.

A client application is required to operate jobs on supercomputers in AEGIS. We developed a client API as a function of AEGIS to develop grid-enabled applications on a client PC. The main functions were implemented on the client API, so users can easily develop their applications using the API. The client API can work on other grid middleware, such as UNICORE, DIET, and Globus.[10] During authentication, we use an IC card or USB e-Token, which includes a certificate (PKCS#11).

The client API is classified into low, middle, and high level APIs based on their functions. In low level APIs, the authentication API, file transfer API, job submission API, and job information API are classified. The jobscript generator API is classified as a middle level API.

Low level APIs supply authentication, connections between a client PC and supercomputers on AEGIS, job operation to supercomputers, resource handling on both the client's PC and supercomputers, and other tasks. The API cooperates with the Remote Install Service (RIS) daemon on computers and requests resource information from comput-

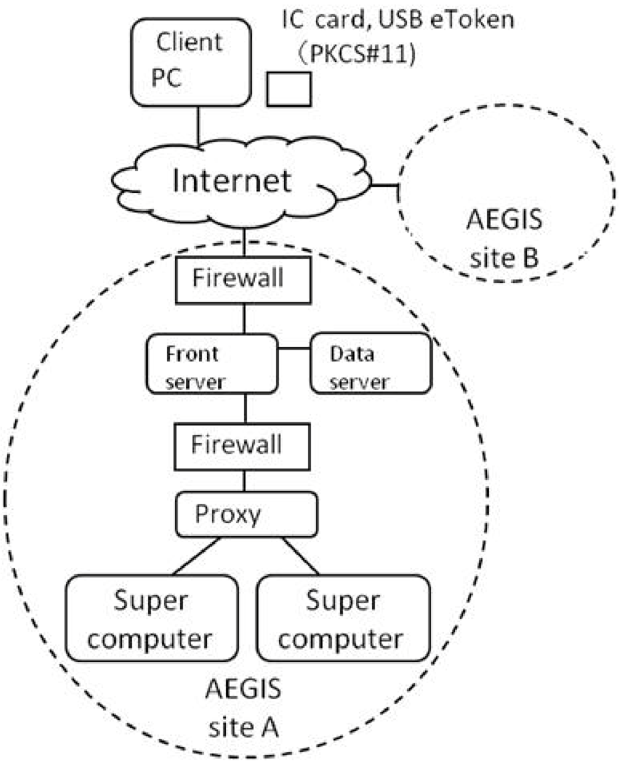

Fig. 1 Schematic diagram of AEGIS

ers, including the job status, at fixed time intervals. For example, if a job exceeds the execution time limit or the job causes an error, such as an I/O error, the AEGIS middleware on the supercomputer replies with an abnormal error. If the communication between the client PC and the supercomputer is disconnected or the middleware on supercomputer stops unexpectedly, the API receives no response. The API then returns “connection error” to the client application. Finally the client application decides that an unexpected outage has occurred.

The jobscript generator API generates a script corresponding to heterogeneous computers by reading job attributes, such as computer name, job class, number of CPUs, path of program, and work directory.

Using the functionality of low level and middle level APIs, the FT method is achieved.

2. Simple Orchestration Application Framework

During integrated simulations we observed the file flow between codes. The file transfer triggers new code execution during the cooperative execution of codes. Therefore, describing the file flow between codes allows us to construct scenarios for integrated simulations.

In addition to the low level and middle level APIs, we also developed SOAF[2,3] as an FT method for job execution and job cooperation. Integrated simulation is driven by file flow. In SOAF, we categorize simulation codes into the following two types: the first is the initial code and the second is the code initiated after receiving files from other codes. At the end of file output or file transfer a "flag file" is created as

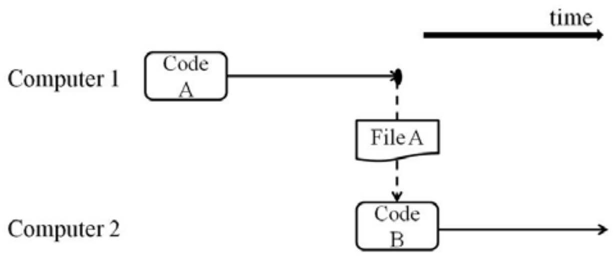

Fig. 2 Basic example. Code A and Code B are executed sequentially. The dashed line indicates file transfer. File A is transferred between code A and code B.

well as the output file or transferred file. SOAF detects the flag file and commences file transfer or job execution.

SOAF consists of a client API (referred to as SOAF API) and server script programs (sentinels). SOAF API is categorized as a dominance layer. When the SOAF API detects abnormal job errors, the API resubmits the jobs to the same or alternative computers via low level APIs and a jobscript generator API. Users can easily implement the FT mechanism on the client application.

We observed the behavior of SOAF. For example, we assumed a basic case (**Fig. 2**). Here we suppose that there are two codes, Code A and Code B. Code A and Code B are executed by Job A on computer 1 and Job B on computer 2, respectively. After the completion of Job A, File A is an output from Code A and it is transferred from Computer 1 to Computer 2. After file transfer, Job B starts and code B reads File A.

Users can operate this cooperative execution from a client PC. Users define the file flow between Code A and Code B. When Code A generates file A, SOAF transfers it from computer 1 to computer 2. After this, SOAF submits the job of Code B.

By defining the file flow, we can execute various integrated simulations using different flow types or categories, such as sequential, concurrent, or branch conditional. The file transfer triggers code execution. The jobs and file transfers are related and defined in the client application. The timing of file transfer is also defined in the client application.

When the SOAF API detects abnormal job errors, the API resubmits the jobs to the same or alternative computers via low level APIs and a jobscript generator API.

SOAF recovers any abnormal job terminations, but it does not handle file transfer errors. The large-scale simulation involves long-term simulation and large file transfers. If a network outage occurs during file transfer, the file transfer will not complete correctly. Thus, we developed a method for long-term cooperative simulations.

III. Fault-Tolerant Mechanism for File Transfer

1. File Transfer Procedure

If a network outage occurs during file transfer, the files will not be transferred correctly. File transfer errors occur during abnormal terminations of integrated simulations. During long-term simulations, the FT method must ensure job execution and correct file transfer.

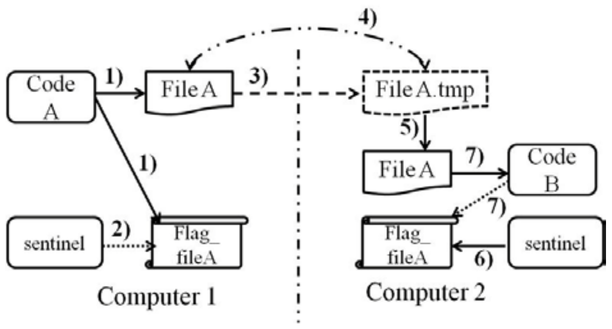

Fig. 3 File transfer between Code A and Code B. After file transfer, the FT API compares the size of File A before and after transfer.

We developed FT API for AEGIS. The FT API monitors the job status and file transfers via low level APIs. The FT API evaluates file transfer between codes by comparing file sizes.

The sentinel monitors file outputs from simulation code. When the sentinel detects a flag file from an output file, the sentinel ends. The client application detects the end of the sentinel and begins file transfer. Cooperative execution requires the installation of a sentinel program in addition to the executable codes.

Here we describe the procedure for file transfer between codes. The procedure for the simple case in **Fig. 2** is shown in **Fig. 3**.

1) Code A generates File A and its flag file.
2) The sentinel for Code A detects the flag file and ends.
3) File A is transferred from computer 1 to computer 2.The name of File A is changed to a temporary name in computer 2.
4) The FT API compares the size of files on computer 1 and computer 2. If these coincide, the API decides that the file transfer has completed correctly. If not, the API transfers File A again until the maximum limit of file re-transfers.
5) File A is renamed to its original name.
6) A flag file for File A is created by the sentinel on computer B.
7) Code B detects the flag file for File A and reads File A.

When the job is stopped by an abnormal error, the low level AEGIS API detects this error through the response to the inquiry received on the computer. The job is then migrated to another supercomputer by the FT API. The relationships between jobs and file transfers are defined in the client application, so the file transfer would change at job resubmission.

If a transfer error occurs, such as a network outage, the low level AEGIS API detects this error via the response to the file transfer command. The FT API retries the transfer, regardless of the type of error, because low level AEGIS

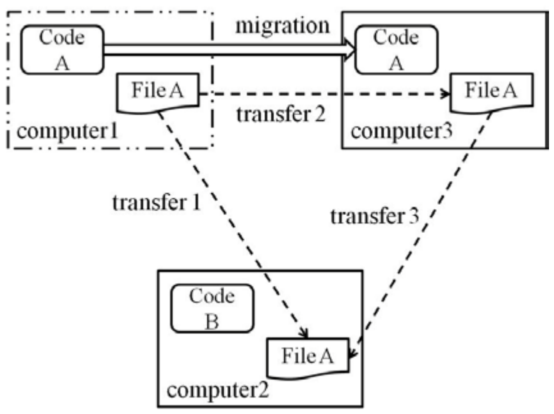

Fig. 4 Migration of code A. Following migration, the FT API changes the file transfer from "transfer 1" to "transfer 2" and "transfer 3".

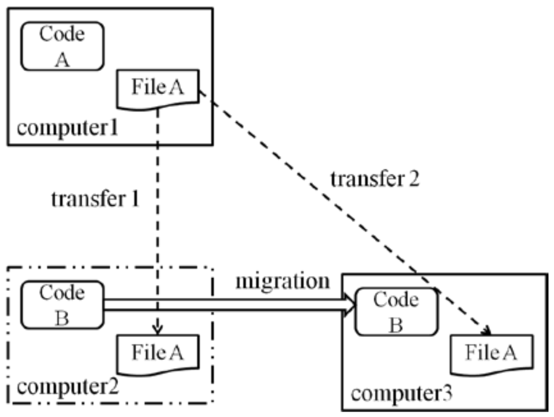

Fig. 5 Migration of code B. Following migration, the FT API changes the file transfer from "transfer 1" to "transfer 2".

APIs cannot determine the origin of the error from the details. The file size is investigated by low level AEGIS APIs on both computers.

We consider that there are two patterns of migration. First, we consider an unexpected outage during job A (**Fig. 4**). If job A is migrated from computer 1 to computer 3, the source of the file transfer is also changed. If computer 1 does not cause outage, the FT API transfers file A in "transfer 2". Then file A is transferred in "transfer 3".

If there is an unexpected outage during job B, the FT API changes the receiver of the file transfer (**Fig. 5**). If job B is migrated from computer 2 to computer 3, the destination of the file transfer also changes. The FT API changes the file transfer from "transfer 1" to "transfer 2".

We now consider a more complicated cooperative execution. Here we suppose that there are 5 codes (code A, B, C, D, and E) running on 5 computers (computer 1, 2, 3, 4, and 5). These codes are executed sequentially as shown in **Fig. 6**. When the execution of code E finishes, connections D-E and E-C are disconnected simultaneously.

We consider two possible cases. If computer 5 is disconnected from all other computers, the low level API on client PC receives no response to a job status query from computer 5. Therefore, the client application decides that an "abnormal error occurred on computer 5." Code E is then migrated to an alternative computer by the FT API. Following this migration, files from code D are also migrated from computer 4 to an alternative computer, according to the configuration on the client application. If no alternative computer is available for computer 5 in the configuration of the client application, the FT API stops all jobs and causes an abnormal termination.

However, if connections between computer 5 and other computers are normal, the file transfer would still not work well. This is because the low level API receives a normal job status response from computer 5, but the file transfer procedure returns an error code. The APIs cannot recognize origin of this error from the details, so the FT API repeats the retransfer from E to C until the maximum number of executions defined in the client application configuration.

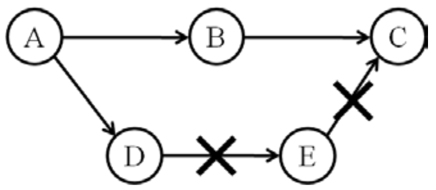

Fig. 6 Cooperative execution of 5 codes. During execution, connections D-E and E-C are disconnected simultaneously.

The FT API cannot recover the file transfer between computer 5 and computer 3. Finally, the FT API returns an error stating that "number of file retransfers is exceeded". The client application then causes an abnormal termination.

2. Evaluation of the FT Method

We evaluated the robustness of file transfer. Here we consider the simple cooperative operation from the previous section (**Fig. 2**). Code A was implemented on a PC cluster at Naka Fusion institute of JAEA. Code B was implemented on p690 at CCSE/JAEA Ueno site. File A was about 15 MB and it was transferred from Naka to Ueno in 15 seconds. After transfer, the FT API compared the file size before and after transfer and changed the name of file (Procedure 4 in previous subsection). This procedure was finished in one second.

We evaluate the behavior after an abnormal error caused by an artificial procedure. In our experiment, the FT API immediately recovered the error. First, we deleted the temporary file for File A. After completion of the transfer procedure on computer A, the FT API detected the absence of File A. The FT API then retried the transfer of File A. The sequence of the procedure from the completion of file transfer to retransfer took only a few seconds. Next, we changed the temporary file for File A to another file before comparing the file size. Of course, the file sizes are difference. After completion of the transfer procedure on computer A, the FT API compares the file sizes. The FT API decides that a transfer error has occurred and retries the transfer. The sequence of procedures from the completion of file transfer to retransfer takes only a few seconds.

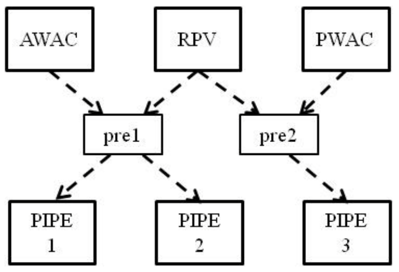

Fig. 7 Cooperative execution of FIESTA. The integrated simulation consists of three layers. The codes for AWAC, RPV, and PWAC are executed in parallel.

To detect file collapse, there is often a comparison of hash values before and after file transfer. We computed hash values with MD5 and SHA1 to test the robustness of file transfer. The time required was as follows: 0.14 seconds on PC cluster, MD5; 0.19 seconds on p690, MD5; 0.17 seconds on PC cluster, OpenSSL 0.9.6b+SHA1; 0.11 seconds on p690, OpenSSL 1.0.0a+SHA1. Our framework can evaluate the actual speed of file transfer, but it cannot detect file collapse. We plan to add a hash function to the framework to allow the detection of file collapse.

IV. Application of the FT API to Nuclear Energy Simulations

We applied the FT API to nuclear energy simulations. Here we consider two integrated simulation scenarios. We used test codes in the evaluation. The relationship between simulation codes and input/output files was defined in the client application configuration for the FT API. The client application operates job execution and file transfer timing based on this definition.

We considered a full-scale three-dimensional vibration simulation for the entire NPP,[1)] which is a full-scale seismic analysis system to analyze the response of an entire digitalized NPP to an earthquake or vibration effect.

The simulator divides the NPP into several component units. Each component was analyzed by FInite Element STructural analysis for Assembly (FIESTA). The mutual vibration effects among components are modeled as data exchanges at component boundaries, at each simulation time step.

We aimed to perform a full-scale seismic response analysis of the High Temperature engineering Test Reactor (HTTR)[15)] at O-arai R&D center of JAEA. In this scenario, the plant consisted of Reactor Pressure Vessel (RPV), Auxiliary Water Air Cooler (AWAC), Pressurized Water Air Cooler (PWAC), and 3 pipes (PIPE1, PIPE2, and PIPE3). In the integrated simulation, the codes for AWAC, RPV, and PWAC were executed in parallel. During the simulation, the boundary conditions among the large components (AWAC, RPV, and PWAC) and the pipes are sent from the codes for AWAC, RPV, and PWAC to those for the pipes. Data conversion codes (pre1 and pre2) were executed among the codes of the large components and the pipes (**Fig. 7**).

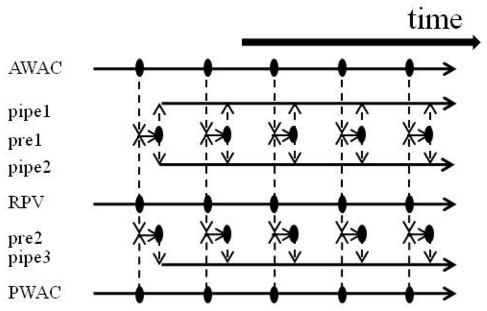

Fig. 8 File flow between simulation codes. AWAC, RPV, and PWAC send boundary conditions among themselves and the pipes. The conditions data is converted by pre1 and pre2. After conversion, data are sent to the pipes. In this integrated simulation, 6 simulation codes are executed in parallel.

File flow between the codes is shown in **Fig. 8**. Six codes were executed in parallel during the integrated simulation. In the client application, we defined the timing of the execution of pre1 and pre2 after file transfer from the large components to the pipes. These codes are executed after pre1 or pre2 received a file from the large components. The output files for the pipes were then generated. The codes of pipes await the receipt of a file from pre1 or pre2. The integrated simulation was conducted after we modified the codes for the pipes relating to the waiting files from the pre codes.

The difference in total execution time between the implementation of the FT API and the SOAF API was less than one minute. The actual execution of this simulation required about ten days, so the difference was tiny.

We now consider another scenario. Simulations for Predicting Quake-Proof Capability of Nuclear Power Plant are considered important for the verification of the safety capacity of ageing nuclear power plants that might be subjected to a large earthquake. An integrated analysis of seismic waves, structure of the nuclear power plant, thermal hydraulics, and the nuclear reaction was required for verification.

The integrated simulation consisted of the Macro-Micro Analysis method (MMA),[16)] ADVanced ENgineering analysis Tool for Ultra large REal world (ADVENTURE),[17)] BWR 3D-Neutronic Thermal-Hydraulic Code (TRAC-SKETCH),[18)] Advanced Code for Evaluation of 3-Dimensional constitutive equations in a two-fluid model (ACE-3D),[19)] and data conversion codes to use among the 4 simulation codes (**Fig. 9**). The codes were executed sequentially.

File flow between the codes is shown in **Fig. 10**. In this scenario, the simulation codes were executed sequentially. In the client application, we defined the timing of the execution of ACE-3D after file transfer from both ADVENTURE and TRAC-SKETCH. During sequential execution, we had to modify the simulation codes to allow generation of a flag file.

A maximum of six stages of file transfer occurred from MMA to ACE-3D. The total execution time for file comparison by FT API was less than one minute. The effect of the file comparison to the total time for executing the simulations was quite tiny.

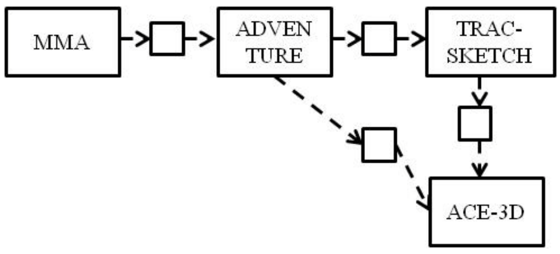

Fig. 9 Cooperative execution of a simulation for Predicting Quake-Proof Capability of Nuclear Power Plant. The codes were executed sequentially. Data conversion codes (small white boxes in this figure) were executed among the 4 simulation codes.

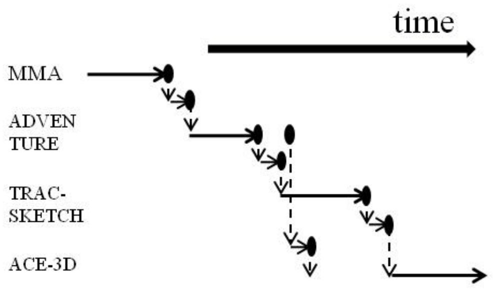

Fig. 10 File flow between simulation codes. The name of the data conversion code is omitted. In this scenario, simulation codes were executed sequentially. ACE-3D executed after receiving files from ADVENTURE and TRAC-SKETCH.

V. Conclusion

We developed a fault-tolerant (FT) method for job execution and file transfer. The FT method was applied to long-term simulations to avoid unexpected computer stoppages or network outages. We implemented the method with an AEGIS API. Users can easily develop client applications that operate in long-term integrated simulations.

In our experiment, the FT method immediately detected file transfer errors. When the transferred file was deleted or replaced by another file, the FT API detected a file transfer error and retransferred the same file in several seconds. We can conclude that the FT method was beneficial for the operation of integrated long-term simulations.

We evaluate the behavior of integrated simulations using the FT mechanism. We considered two scenarios from the nuclear energy field. In the first scenario, simulation codes were executed sequentially. In the second scenario, the codes were executed concurrently. In both cases, the client application could operate the integrated simulations. We controlled various simulations of the nuclear system using the FT method.

We considered the operation of realistic nuclear energy simulations. For example, execution of the three-dimensional Virtual Plant Vibration Simulator required about 10 days for 99 steps. Using the FT API, we could recover job execution and file transfer in long-term simulations. Substantial modification of codes was not needed for cooperative execution. Use of the FT API enables the automated execution of integrated simulations and is also expected to reduce the development time for integrated simulations.

Acknowledgments

We thank R&D team for full-scale simulation technology at JAEA/CCSE and O-arai R&D center of JAEA for useful discussions. This work was supported by the Japan Science and Technology Agency (JST/CREST) "Simulation for Predicting Quake-Proof Capability of Nuclear Power Plants" project.

References

1) Y. Suzuki, A. Nishida, F. Araya, N. Kushida, T. Akutsu, N. Teshima, K. Nakajima, M. Kondo, S. Hayashi, T. Aoyagi, N. Nakajima, "Development of Three-dimensional Virtual Plant Vibration Simulator on Grid Computing Environment ITBL-IS/AEGIS," *J. Power Energy Syst.*, **3**[1], 60–71 (2009).
2) T. Tatekawa, K. Nakajima, G. Kim, C. Kino, N. Teshima, Y. Suzuki, H. Takemiya, *Cooperative Execution Framework for Integrated Simulations on Grids*, FUJITSU Family Association FY2009 (2010), [in Japanese].
3) T. Tatekawa, K. Nakajima, G. Kim, N. Teshima, Y. Suzuki, H. Takemiya, N. Hayashi, K. Iba, "Simple Orchestration Application Framework to Control "Burning Plasma Integrated Code"," *Proc. of The Third International Joint Conference on Computational Sciences and Optimization (CSO2010)*, The 2010 International Workshop on HPC and Grid Applications (IWHGA2010), Huangshan, Anhui, P. R. China, May 28-31, 2010, **2**, 322–326 (2010).
4) B. Ludäscher, I. Altinta, C. Berkley, D. Higgins, E. Jaeger, M. Jone, E. A. Lee, J. Tao, Y. Zhao, "Scientific workflow management and the Kepler system," *Concurrency Comput. Pract. Ex.*, **18**, 1039–1065 (2005).
5) T. Imamura, Y. Hasegawa, N. Yamagishi, H. Takemiya, "TME: A Distributed resource handling tool," *Recent Advances in Computational Science & Engineering, International Conference on Scientific & Engineering Computation (IC-SEC)*, Dec. 3-5, 2002, Raffles City Convention Centre, Singapore, 789–792, (2002).
6) K. Seymour, H. Nakada, S. Matsuoka, J. Dongarra, C. Lee, H. Casanova, "Overview of GridRPC: A Remote Procedure Call API for Grid Computing," *Proc. of 3rd International Workshop on Grid Computing*, M. Parashar (ed.), 274–278 (2002).
7) T. Imamura, Y. Tsujita, H. Koide, H. Takemiya, "An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers," *Proc. of the 7th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, 200–207 (2000).
8) I. Fostar, N. T. Karonis, "A grid-enabled MPI: message passing in heterogeneous distributed computing systems," *Proc. of the 1998 ACM/IEEE conference on Supercomputing*, 1–11 (1998), [CD-ROM].
9) T. Beisel, E. Gabriel, M. Resch, "An extension to MPI for distributed computing on MPPs," *Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes in Computer Science*, **1332**, 75–82 (1997).
10) Y. Suzuki, N. Kushida, N. Teshima, K. Nakajima, A. Nishida, N. Nakajima, "Atomic Energy Grid InfraStructure (AEGIS) and Interoperation with Other Grids," *High Performance Computing on Vector Systems 2008*, M. Resch, S. Roller, K. Benkert, M. Gelle, W. Bez, H. Kobayashi, T. Hirayama (Eds.), Springer-Verlag Berlin Heidelberg, 65–77 (2008).

11) G. Kim, Y. Suzuki, N. Teshima, “Network Computing Infrastructure to Share Tools and Data in Global Nuclear Energy Partnership,” *J. Power Energy Syst.*, **4**[1], Special Issue on 17th International Conference on Nuclear Engineering 180–190 (2010).
12) Y. Suzuki *et al.*, ”Research and Development of Fusion Grid Infrastructure Based on Atomic Energy Grid InfraStructure (AEGIS),” *Sixth IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation for Fusion Research*, Jun. 4-8, 2007, Inuyama, Japan, *Fusion Eng. Des.*, **83**, 511–515 (2008).
13) A. S. Tanenbaum, M. V. Steen, *Distributed Systems: Principles and Paradigms (2nd Edition)*, Prentice Hall, Upper Saddle River, N.J. (2007).
14) Y. Suzuki, *Research and Development of Application Programming Interface for Grid Environment*, FUJITSU Family Association FY2007 (2008), [in Japanese].
15) S. Saito *et al.*, *Design of high temperature Engineering Test Reactor (HTTR)*, JAERI 1332, Japan Atomic Energy Research Institute (JAERI) (1994).
16) T. Ichimura, M. Hori, “Macro-Micro Analysis Method for Strong Motion Distribution,” J. *Struct. Eng./Earthquake Eng., JSCE*, **I-52:654**, 51–62 (2000).
17) I. Yamasaki, S. Yoshimura, *Environmental Equipment for Usage of FEM software –ADVENTURE System User’s Guide -*, JAERI-Tech 2003–050, Japan Atomic Energy Research Institute (JAERI) (2003).
18) V. G. Zimin, H. Asaka, Y. Anoda, M. Enomoto, “Verification of J-TRAC code with 3D neutron kinetics model SKETCH-N for rod ejection analysis,” *NURETH-9*, San Francisco, California (1999).
19) A. Ohnuki, H. Kamo, H. Akimoto, *Development of Multidimensional Two-Fluid Model Code ACE-3D for Evaluation of Constitutive Equations*, JAERI-Data/Code 96–033, Japan Atomic Energy Research Institute (JAERI) (1996).