氏　　名	Thepparit Banditwattanawong
学位（専攻分野）	博士（情報学）
学位記番号	総研大甲第1053号
学位授与の日付	平成１９年３月２３日
学位授与の要件	複合科学研究科　情報学専攻 学位規則第６条第１項該当
学位論文題目	A study on Fine-grained Replications of Distributed Java Applications
論文審査委員	主　査　教授　丸山　勝巳 教授　中島　震 教授　佐藤　一郎 助教授　児玉　和也 助教授　千葉　滋（東京工業大学）

論文内容の要旨

In distributed object systems, object-oriented (OO) applications are replicated from remote servers to client sites to improve performance, scalability, and availability. This study focuses on fine-grained replications of distributed OO applications. Unlike the traditional replication scheme by which a self-contained application is replicated entirely at once, the fine-grained replication scheme enables partial and on-demand incremental replications of self-contained applications.

Fine-grained replications can be classified into two categories based on their deployment patterns: 1) replicating running applications for local accesses and 2) downloading application programs from persistent repositories for local executions. Based on the classification, the study has proposed a pair of fine-grained replication middlewares: one aims for the fine-grained replications of remote runtime applications, and the other aims for the partial and on-demand incremental downloadings of application programs.

In addition, to exploit the fine-grained replications effectively requires a proper means to figure out application portions as the units of replication. The study has proposed object class clustering algorithms to support the use of the latter middleware, while showing that object clustering, which is used to support the former middleware, can be performed based on programmer's application knowledge.

The details of the middlewares and the class clustering algorithms are summarized individually as follows.

Fine-grained replication of runtime application:

Replicating remote application objects to user locality is a common technique to reduce the effects of network problem. The traditional replication scheme is not suitable for cooperative applications because only part of a shared application rather than a whole application should be replicated. Furthermore, the scheme is not appropriate for mobile computing devices due to their common constraints of memory spaces. Both problems can be addressed by using a fine-grained replication scheme by which the portions of a self-contained application can be replicated.

Since most object replication systems exploit the traditional replication scheme, to fulfil fine-grained replication is an unexperienced task for several application programmers. There exist few middlewares that support runtime fine-grained replications of OO applications. All of them aim for peer-to-peer applications in

which objects that constitute a self-contained application are decomposed and distributed among peers. Therefore, peers that hold master copies of the application objects must always be reachable by other peers to replicate the master copies. This is not suitable for pervasive collaboration because the servant peers (e.g., mobile users) can get disconnected arbitrarily or be unreachable due to network partitioning. Instead, using dedicated servers to maintain the master copies of applications is more appropriate. Unfortunately, no fine-grained replication middleware is designed for a client-server model.

This dissertation presents SOOM, a Java-based middleware for pervasive client-server cooperative applications. SOOM provides fine-grained replication capability for clients in wide-area networks or on the Internet and allows clients in local area network to exploit a conventional remote method invocation mechanism in coordination with the fine-grained replication. SOOM also supports fine-grained concurrent access control and update synchronization. To realize the middleware, several challenge research problems have been identified and resolved.

An application for cooperative software modeling has been developed to assure the practical applicability of SOOM and demonstrate the practicality of fine-grained replication scheme, fine-grained consistency maintenance, and the coexistence of fine-grained replications and remote method invocations in client-server environment. The quantitative properties of SOOM were measured through the following empirical evaluations. First, experiments in single-user and multi-user environments based on different consistency protocols indicated the practical throughputs of SOOM-based application. Second, throughout accessing all member objects of a benchmark cluster showed that SOOM-based replication began to outperform Java RMI when each object was accessed locally more than twice. Third, an experiment using the varied numbers of client processors assured the scalability of SOOM. Finally, an experiment on the memory space requirement showed that SOOM could reduce the significant amount of client memory space consumption as well as network bandwidth.

Fine-grained replication of application program:

OO applications have been distributed more and more over the Internet. Deploying an application by retrieving the entire program from a remote repository such as HTTP server often encounters extended delay due to network congestion or large program size. Many times system resources, such as network bandwidth and client memory space, are also wasted because users do not utilize every component of the downloaded applications. Moreover, downloading a whole program at once is usually impractical for mobile computing devices due to their memory space constrains.

These problems can be addressed by decomposing a program into groups of classes and data resources to be downloaded on demand.

This dissertation presents C^2, a Java-based middleware by which a Java application can be partially and on-demand incrementally deployed via HTTP. The middleware also supports application caching and transparently automatic updating.

The launching delay of an experimented application was found to be reduced by 83% from that of the traditional whole-at-once application deployment scheme. Total program deployment and execution overhead was 22% less than that of Java Web Start.

Object class clustering approach:

It is typical that only part of whole program code is necessary for successful execution. Decomposing an OO program into clusters of closely relevant classes and data resources for on-demand incremental loading optimizes the program start-up latency and system resource consumption. The lack of systematic yet simple class clustering approach prohibits this kind of optimization.

This dissertation presents a Java class clustering approach that is capable of improving both spatial locality and temporal affinity of the optimized programs. The approach provides two clustering algorithms: initial delay-centric and intermittent delay-centric ones, to achieve different requirements of optimizations.

Experimental results indicated that the algorithms were practically useful to both interactive programs and non-interactive programs. Among the tested Java programs, using the initial delay-centric algorithm and the intermittent delay-centric algorithm improved initial program loading latencies on average by 2.9 and 2.2 times respectively faster than the traditional whole-at-once program loading scheme. The intermittent delay-centric algorithm reduced the number of intermittent delays to half of the initial delay-centric algorithm. Both algorithms also led to the chances to economize on system resources, such as memory spaces and network bandwidths.

論文の審査結果の要旨

本論文は、JAVA分散オブジェクトの効率やScalability向上のための部分的複製とその実行に関する研究である。例えば、遠隔地にいる人々がCSCW (Computer Supported Cooperative Work) 環境で一つのソフトウェア開発をしている場面を考えよう。中央のサーバでは開発中のシステム全体が動作している。各プログラマのクライアントマシンには、彼が開発担当している部分プログラムと参照先の部分プログラムの複製が作られ動作するので、担当者間で部分プログラムの重なりが少なければ、効率が改良される。また、部分的にしか複製を作らないので、メモリや通信速度に制限の強い携帯機器を使った共同作業にも有効となりうる。

第 1 章は、本研究の背景と必要性について述べている。Client/Server 型分散オブジェクトシステムの性能向上の課題をサーベイし、クライアントマシン上にサーバー上のプログラムの複製を作ってローカル実行すること; 複製はプログラム全体ではなくクラスタ単位(ここに、クラスタとはプログラムのサブグラフで、一緒に実行される可能性が高い固まりを言う)が適切なこと; を述べている。その実証研究として、クラスタ単位での複製・実行(第 2 章), クラスタ単位でのプログラム配信(第 3 章)、及びクラスタ分けアルゴリズム(第 4 章)の研究への繋ぎを述べている。

第 2 章は、クラスタ単位の複製・実行の手法とその実装 SOOM (Scalable OO Middleware) について述べている。遠隔サーバマシン上の JAVA 分散オブジェクトプログラムに対し、クライアントマシン上にプログラム実行全体ではなく必要に応じてクラスタ毎に複製を作って実行させる手法とその実装 SOOM である。クライアント側でクラスタが参照されたらサーバー側と通信して自動的に複製を作る Proxy-hook 機構、クラスタ間のリンクを管理するクラスタテーブル、及びきめ細かい同期制御によるという面で、独自性が認められる。

第 3 章は、クラスタ単位のプログラム配信と、その実装 CC(Class cluster replication middleware)について述べている。携帯端末など通信速度が劣る環境では、大きなプログラムをサーバーからダウンロード・立ち上げに時間がかかる。本手法では、実行に必要なクラスタだけを実行時にロードするので、低品質 NW 環境下では立ち上がり時間を短縮でき、メモリ等のリソース消費も抑えられることを、システム CC を試作し、評価で示している。

第 4 章は、オブジェクト指向プログラムのクラスタ分割アルゴリズムについて述べている。基本的な発想は、プログラムのフローグラフを分析して、確率的にまとまって実行されるであろう部分毎にクラスタ化することにある。実用に供するにはループ実行回数の予測など動的属性も含める必要があるが、一つの研究視点の提案と言える。

第 5 章は、本論文の結論と展望を述べている。本手法は全応用域で効果を発揮するものではないが、巨大だがクラスタ間の結合度が低いような応用域では有効である。

以上のように、本論文の分散オブジェクトの複製は、汎用解ではないが、想定分野では有用な独自手法の提案といえる。研究成果は、学術論文誌並びに国際会議で発表しており、また試作したプログラムはソースコードも含めて、
WEB http://research.nii.ac.jp/H2O/soom で公開している。
以上より、本論文は博士 (情報学) の学位論文として価値があるものと認められる。