Future Generation Computer Systems 74 (2017) 199–207

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Optimized dependent file fetch middleware in transparent computing platform

Kehua Guo [a,b,*], Yayuan Tang [a], Jianhua Ma [c], Yaoxue Zhang [a]

[a] School of Information Science and Engineering, Central South University, Changsha, China
[b] Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education (Nanjing University of Science and Technology), Nanjing, China
[c] Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan

HIGHLIGHTS

- A middleware solution is proposed to optimize file fetch process in transparent computing (TC) environment.
- The method solves the concurrency control problem when the file fetch is required for the multiple clients.
- We propose a heuristic and greedy (HG) algorithm.
- HG algorithm can reduce overall file fetch time roughly by 50% in the best cases compared with the time cost of traditional approaches.

ARTICLE INFO

Article history:
Received 24 August 2015
Received in revised form
17 September 2015
Accepted 14 October 2015
Available online 17 November 2015

Keywords:
Middleware
File dependency
File fetch
Heterogeneous network
Transparent computing

ABSTRACT

A middleware is proposed to optimize file fetch process in transparent computing (TC) platform. A single TC server will receive file requests of large scale distributed operating systems, applications or user data from multiple clients. In consideration of limited size of server's memory and the dependency among files, this work proposes a middleware to provide a file fetch sequence satisfying: (1) each client, upon receiving any file, is able to directly load it without waiting for pre-required files (i.e. "receive and load"); and (2) the server is able to achieve optimization in reducing overall file fetch time cost. The paper firstly addresses the features of valid file fetch sequence generating problem in the middleware. The method solves the concurrency control problem when the file fetch is required for the multiple clients. Then it explores the methods to determine time cost for file fetch sequence. Based on the established model, we propose a heuristic and greedy (HG) algorithm. According to the simulation results, we conclude that HG algorithm is able to reduce overall file fetch time roughly by 50% in the best cases compared with the time cost of traditional approaches.

1. Introduction

In the past decades, the computing pattern has gradually evolved from the desktop terminals to heterogeneous distributed computing. Researchers applied heterogeneous distributed computing to various research areas, including software adaptation [1,2], E-Commerce [3] and heterogeneous multimedia big data retrieval [4], and proposed many achievements. Traditional view of computing is hardware or software-centered, now it is slowly turning into service-oriented [5]. Transparent computing [6,7] is a concrete application form of heterogeneous distributed computing, and has become an emerging computing pattern which can prevent users from worrying about the heterogeneous software details (e.g. operating systems (OSes), supporting tools and applications (Apps) which are stored and managed in centric servers), and can satisfy their service needs through simple interfaces supported by heterogeneous distributed devices. That is to say, the client will not store any OSes and Apps locally. Instead, they will be downloaded from the server according to user's request in the client interface. The goal of this computing pattern is to reduce the burden of the server and improve the system maintainability. Therefore, users simply just accomplish their tasks on their data and do not care about the machine specifics and management details [8].

In transparent computing platform, one server is usually responsible for the file transmission requests from multiple het-

* Corresponding author at: School of Information Science and Engineering, Central South University, Changsha, China.
E-mail address: guokehua@csu.edu.cn (K. Guo).

http://dx.doi.org/10.1016/j.future.2015.10.010
0167-739X/© 2015 Elsevier B.V. All rights reserved.

erogeneous distributed clients. The transmission process between server and clients has a very important place in the research of transparent computing. For the TC clients, requesting various files to support applications and OS is a very common job. Due to the diversity of files requested (such as operation system, application or user data), file transmission in TC has encountered a special challenge—the dependency among these files. In many cases, to successfully run heterogeneous applications or operating systems, files must be loaded in a specific order. For example, when a voice-text input application is requested by a client, the client must firstly load the voice input/output module and then the voice-text translation module. If the voice-text translation module is firstly received by the client, it will be held until the voice input/output module is transferred. In another case, a client requests a text-voice reader application, the dependency order could be reversed. All the files are fetched from the server, when multiple heterogeneous clients request various files. It is an important issue to generate a reasonable file fetch and transmission sequence at the server.

To solve the file dependency problem, a middleware can be developed to schedule the fetch sequence, which will bring critical influence upon heterogeneous distributed clients concerning the Quality of Service (QoS) and also improve the efficiency of TC servers. On the one hand, the client is greatly expected that the files can be loaded immediately after the transmission is completed, instead of being held due to some uncompleted transmissions of those files it relies on. If too many files are held at the same time, the client will waste time and hardware resources, which will lead to huge drop in the performance, especially to some devices with the limited computing ability and memory size. On the other hand, multiple heterogeneous distributed clients may request the same files from the single server. It will waste considerable time to repetitively fetch(search and read) the same files for each clients. In consideration of the fact that there are huge overlap among users' requests for the files in TC (e.g. OS files), saving file fetch time will definitely promote the server's performance. In our model, with a file fetch sequence and a fixed size memory buffer in the middleware, the best case expected is to fetch each identical file one time and send it to multiple clients. Thus, the server should provide an optimized file fetch sequence which satisfying: (1) each client, upon receiving any file, can directly load it without waiting for pre-required files (i.e. "receive and load"); and (2) the server can achieve optimization in reducing overall file fetch time cost. These requirements have become new challenges in the area of file transmission in transparent computing platform.

In recent years, the research of TC has been more and more popular. Our research group is performing some important research issues. In cooperation with scholars from other universities, we are constructing an optimized algorithm to dependent file fetch middleware in transparent computing platform. This paper proposes the key definitions and features of the dependent file fetch problem, describes a heuristic and greedy algorithm in the middleware, and finally evaluates the performance of the proposed algorithm.

The rest of this paper is organized as follows. Section 2 explains the related work of transparent computing and file transmission approaches, Section 3 describes the system model and problem description, and Section 4 presents the solution for the problem, including all the related information, algorithm description and complexity analysis. Section 5 provides performance evaluation and experimental results. Section 6 concludes our work and points out the future work.

2. Related works

Transparent computing is a computing paradigm to provide transparent services for users [9]. In TC architecture, users only need to care about the result and quality of the services they want, and do not have to pay attention to the details of the Apps in system. Therefore, users are able to freely access the services on network across heterogeneous software and hardware platforms [10]. The detailed illustration of TC architecture is shown in Fig. 1.

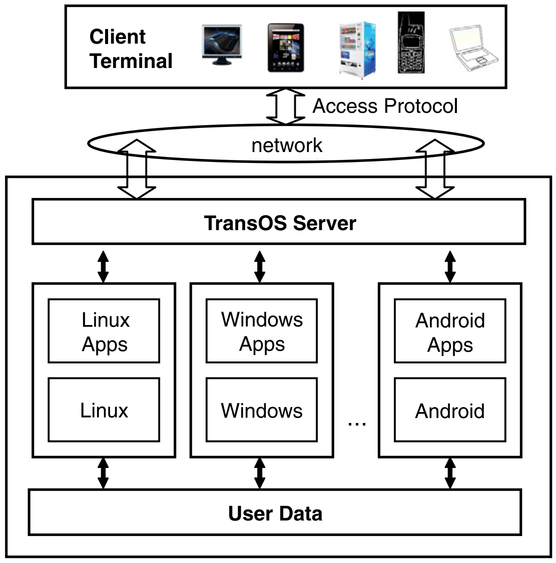

Fig. 1. Transparent computing architecture.

As shown in Fig. 1, a TC based system mainly consists of two units. One is TransOS (Transparent Operating System) server. TransOS server is a network server which stores OSes, Apps and user data. Computation and storage are spatio-temporally separated. When users request some services, the suitable OSes and Apps will be dynamically dispatched from the TransOS server to run on the client terminal in a buffer-enabled block or streaming way. The other is client terminal. In TC paradigm, the client terminal can be diverse and light-weighted; it only needs to store the underlying Basic Input Output System (BIOS) and a small fraction of protocol and management program. For this reason, the client can be securely and easily managed and maintained. Therefore, in TC architecture, computation, storage and management are separated. The users can access the services through cross-terminal and cross-OS operations [11].

The concept of TC was proposed in 2004, Ref. [12] gave the concept, architecture and example of transparent computing. In the past decade, based on the theory of TC, researchers proposed some representative architectures and developed demonstration applications. Ref. [13] developed the client terminal and related systems according to their TC study and proposed a novel Meta OS approach for streaming programs named 4VP. Ref. [14] proposed the performance modeling and analysis algorithm of the booting process in TC environment. In 2012, Ref. [15] reported the work on building a virtual machine-based network storage system for TC platform.

From 2007, a cooperated research team was established with Intel Corporation by combining a new-generation BIOS named UEFI (Unified Extensible Firmware Interface) with TC architecture [16,17], many applications were proposed based on this combination platform. Some research groups have developed TransOS clients supporting various client hardware architectures (e.g. x86, ARM and MIPS) through UEFI [7].

For the transmission protocol, Kuang et al. developed a novel network storage access protocol for TC named NSAP [18]. For the virtualization technology, Refs. [6,10] presented a separating computation and storage strategy. In addition, Refs. [7,19] described the relationship between transparent computing and cloud computing.