Mobility of Parallel Processing

Mark Odell
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
mark.odell@sjsu.edu

1. Abstract

In this paper I discuss and summaries the classifications and characters of Distributed Application Processing Frameworks or DAPFs and apply them to two particular case studies. In particular, I explain the differences in Virtual Machine, Entire Application and Application Partitioning Models for offloading as well has server granular characteristics of offloading as discussed in " A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing". I apply these characteristics to a autonomous system and parallelize web browser. Finally, my discussion makes predictions on the future may hold for Mobile processing as well as what they provide today.

2. Introduction

The interactions between computers has created a more power system. Theory about connection of machines facilitates computational power. Things today already use the interconnected computational models. Mobile Processing.

2.1 Mobile Processing

Mobile processing definition. Mobile processing is computing that takes place on computers that have connections that allows them to move location. We know this as wireless communication. Computation can occur on a mobile device and then transported to another host by wireless mediums in a collaborative effort for a common goal or a one way sending of information.

What is means in present society. Some examples. In present times, there is multitude of devices and frameworks that use mobile processing to reach many different desired effects. These include cell phones, laptops, iPods and other media players. Most if not all of these devices use a 'Cloud' based framework to support and maintain information on demand. That is that the client mobile devices request information from a host server when it needs it and may put some information back on the host server. It is a well know and used computational distribution model that allows mobile devices to have produce results computations that can be extremely intensive or otherwise impossible on a small mobile device.

Lead in distributed systems. Cloud based frameworks are not the only distributed computational frameworks. Numerous frameworks lend themselves to different conditions, parameters and situations.

2.2 Classification of Distributed systems

These frameworks have been given a acronym: DAPFs or Distributed Application Processing Frameworks [1]. A DAPF is a abstract framework that an application may be built on. It characterized by offloading computation onto other hosts or entities to assist, complete, or compute the entire computation. In this paper I have use the taxonomy and classification style of [1] to identify and label some frameworks that illustrate a DAPF.

The classifications that are used to analyze [2] [3] are explained in Section 2 of this paper. It should be noted that this classifications are not mutually exclusive and can and often do mix. We use these classifications as a way of understanding and identifying computation offloading so that it may be quantified and compared to other frameworks or systems.

2.3 Objectives of Distributed Applications

Objectives of DAPFs are similar to that of any application framework. They include scalability, security, robustness, ease of implementation. However there is an additional perception which distributed frameworks take into account power consumption. This is explained in Section 2.3 of this paper.

3. Defining Mobility of Digital Information

3.1 The Three flavors of Distributed Applications

Underlying commonality is Application Offloading, in varying degrees. But largest classification use in [1] is Virtual Machine, entire application or application portioning offloading. In [1] we can see the level of offloading in

3.1.1 Virtual Machine Migration

Define and Characterize Virtual Machine Migration DAFPs. A virtual machine is running on a mobile device that is offloaded to servers to complete the computation. In a sense it is similar to the Entire Application Migration Model since the entire application is being offloaded to another server but with the caveat that we are passing a Virtual Machine instance instead regular tasks.

Some applications that use the Virtual Machine Migration model include VMWare's vSphere [4] and KVM OS [5] support migration of Virtual Machines.

3.1.2 Entire Application Migration

Define and Characterize Virtual Machine Migration DAPFs. Entire Application Migration Model of Distributed Application Processing Frameworks offloaded their tasks at either at critical points in computation or the application is made so that the intensive computation is offloaded to the server application before runtime. That is to say the application is built with a certain server host in mind that will take over the computation at a given point.

3.1.3 Application Partitioning

Define and Characterize Virtual Machine Migration DAFPs. The Application Partitioning Model indicates a model where the application offloads processing tasks during runtime. A mobile device may send multiple server nodes with some computation

task or the mobile device may request computation done by server nodes. Entire Application and Application Partitioning may seem very similar but the difference to note is that Application Partitioning is done during runtime and makes more flexible in its network parameters.

3.2 Granularity of Parallel Processing

3.2.1 Entire Process

Clearly, Entire Process offloading is dramatic and quite possibly the most excessive use of parallelizing of processing. This occurs in the Virtual Machine migration and Entire Application Models for DAPFs. While is allows for less computation and therefore resources needed on the mobile device is puts that load on the host server and depends on the network connection between the two in order to its computation done. Entire processing has its uses however. We are able to have mobile devices whose battery life will dramatically increase when entire processes are offloaded to a host server.

3.2.2 Module Level

Module level offloading provides a lower level and less server costly model for parallelism in applications. Instead of offloading an entire process when can do modules, or pieces of the application that are related or similar in computation. This allows for continuity between the application and server while still allowing the mobile device make decisions on the application flow. This is seen in Application Partition Models. Module level use can vary for situations.

3.2.3 Method Level

Method Level offloading is another step down in granularity. With this step down it provides a new form to create parallelism and allows for more control on the mobile device. As the granularity of parallelism gets smaller, the ability and accessibility of server nodes to the mobile device increases and can be more utilized. We see frameworks such as surrogate cyber forging or sensing networks which allow for offloading and balancing in a very specific manner.

3.2.4 Thread Level

Thread level offloading is a step up or sorts from Method Level. In Method level offloading we allow a thread to run concurrently during runtime of an application. This is seen in clustering frameworks and cyber forging frameworks. There is less specialization and more load balancing in this type of partitioning.

3.2.5 Weblets

Weblets are a sort of specific application partitioning that allows a continuous computation to occur when being transferred back and forth from mobile devices and server hosts. We see these characteristics in what is called elastic mobile applications [1]. While it provides a maintainable and stable environment whether the application is being run on the cloud server or the mobile device, it limitations are hindered by the mobile and servers' dynamic processing power. That is to say that whichever device is running the application must dynamically determine what kind of device is running the application and adjust accordingly.

3.3 Objectives of Distributed Applications

There are several common objectives that application frameworks in general try to meet and common challenges that appear.

Scalability of a DAPF is most always a concern. As the name implies these frameworks are distributed across multiple hosts and devices and therefore need to be modular enough so that they have the ability to fully utilize all machines for a given application. These scalability depends on the framework and the situation of the application. For Virtual Machine and Entire Process offloading models, the need for scalability is lessen in the fact that these model require less computation for the mobile devices and more so on the server hosts. That is not to say that there may be Virtual Machine frameworks that use Thread or Method offloading to do its computation. These classifications are not mutual exclusive and most certainly combine and mix with any particular application to fit its need.

In any case reliability and availability of central resources is needed and plagues DAPFs[1] from being effective in large networks. The large the DAPF the harder it is to obtain the resources on demand; with exceptions of course. Since DAPFs usually communicate through wireless medium, network volatility is the greatest limiting factor for the mobile applications built on top of these frameworks. In simpler words, DAPFs are only as good as the network they are built one. It is imperative that designers of these DAPFs realize this concept and fully utilize the network topology.

Another hindrance is power consumption on mobile devices. There are DAPFs that use do well at using the smallest amount of battery possible. These networks using include Virtual Machine migration or Entire Process migration. However, it is the goal of all DAPFs to reduce power consumption. It is an implicit trait about offloading processing to other hosts or machines that brings the decrease in power consumption on the mobile device. It should be noted that computation must take place somewhere in order to get the desired result so just like the information is offloaded so is the power consumption. However, with a large enough framework, that is one with many server hosts or mobile devices and assuming the network uses each machine equally, power consumption will be spread between all machines.

4. Intelligent Control of Mobile Robots in Parallel

In the DRTA autonomous robot design [2] the system uses distributed system across several sensors each with their own dedicate processors and control modules, effectively making it a sort of cluster design. The project itself may be dated, 1996, however for illustrative purposes it is an excellent use and example of a DAPF. The robot has modular sensors that communicate to special control units called LICA (Locally Intelligent Control Agent) which is the control unit for each sensor. Some sensors have multiple LICA to assist with conversion of sensor stimulus into digital indicators. Interestingly, there is no one master LICA that controls all the functions of a particular motor capability of the robot. Instead each sensor module is connected via serial ports and communicates via messages sending with headers. This network of modules creates a system of feedback loops that allow each module to make decisions based on a prior program [2].

4.1 Classification

Since the robot uses a series of modules with no particular module acting as a master it is using a locally partitioned distribution network that communicates with each by sending messages over serial ports. Each of the sensory modules use a dedicated partition of the source code to carry out their tasks and processing. Clearly

this system is using module level parallelization, the source code, which was written in Concurrent C [2], is a direct indicator of this.

4.2 Objectives Addressed

This autonomous robot addresses the simple requirements of a DAPF, that is a distributed network that offloads processing to other hosts or control units in this case.

5. Zoomm the Application Partitioner

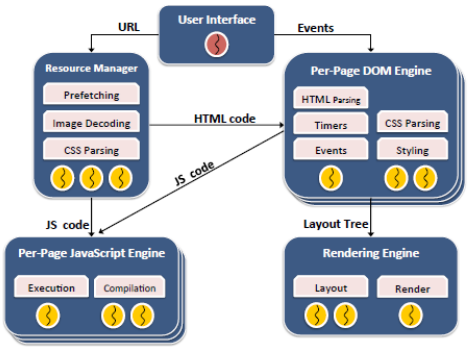

Figure 1: Zoom Architecture as taken from [1].

Zoomm is a parallel browser architecture that allows for a limited but interesting offloading of processing. While the scale of this DAPF is small, it can still be considered a DAPF by the fact that the concurrent functionality this framework provides. Zoomm has several features that allow it to continue processing HTML, image assets, or other resources while it continues to process. Specifically, the browser allows its HTML parser to continue parsing DOM elements even after it has a script reference to another styling sheet or Javascript file. Non-concurrent browsers would have to wait for these assets to load before continuing. The below text is a sample HTML document which we could illustrate the concept of this architecture not having to wait for the five assets that it needs to download. Instead the application would continue past those lines and parse the `title` and `body` tags.

```
<!DOCTYPE html>
<html>
  <head>
    <script src="somesource1.js"/>
    <script src="somesource2.js"/>
    <script src="somesource3.js"/>
    <link rel="stylesheet" href="style1.css">
    <style type="text/css">
      h1
      {
        color: green;
      }
    </style>
    <title>Sample HTML</title>
    </head>
    <body>
      <h1>Everything is Awesome</h1>
      <p>
        When you are part of a team.
      </p>
    </body>
</html>
```

5.1 Classification

This browser uses Application Partitioning and Thread and Process Level parallelism. The Zoomm browser divides its application into several threads and processes. Each dedicated to parse, collect, or download assets as specified by the HTML document. In addition to threads being created a process is created for each webpage that the browser opens. Zoomm has the ability to offloads its application to other cores.

5.2 Objectives Addressed

The Zoom browser addresses the latency when downloading and parsing assets for an HTML document.

6. Discussion

6.1 What all these present technologies provide for us now?

The mobility of mobile applications has changed our interaction with digital media very quickly within the past few decades. We are able to view emails, read social media, view internet sites, view, edit or obtain documents with just a mobile device and even watch television shows while out and about. We have the capability with cell phones edit and whatever digital information we need to edit and save it back into the cloud to be used by yourself or another use at a later time. In addition to making a basic consumer's life more productive, many of the DAPFs have led to more advanced systems such as autonomous robots [2], and other autonomous machine learning systems that have self-sustaining capabilities.

The scalability of these projects has enhanced their growth and influence but most consumer DAPFs computation power is limited by the fact that the application needs collaborate with a few select devices. I am not saying that level of scalability has not change since the beginning of computer history but only that there is still room for much improvement. There is a grander scale on which many frameworks can compute a common goal. This grandeur is seen in frameworks that use a 'crowd sourcing' model of distributed frameworks. Such as SETI@home[6], or Sony's Playstation 3 molecule folding application[7] that uses Standford's Folding project[8]. This crowd sourcing is a form of surrogate-master DAPF that assigns computational problems to given machines to solve and report the result back to the master.

6.2 What I suspect will happen in the future?

Collaborative scaling of distributed applications will continue to grow but at a much slower rate than enterprise scaling. Since not all individual people are willing to give up their computers to random requests for computation to be done, individuals tend to steer away from the idea. Perhaps if the software was presented in a secure and pleasing way more people would be interested in supporting crowd sourcing computation. In addition, if there was an element of competition for individuals to follow and compete for, it would make push people to use the software more. SETI@home[6] has elements of this by having ranks for individuals but that is the extent of the competitiveness. Clearly computational crowd souring is happening very frequently but not to the extent that it could be.

I expect to see more crowd sourcing frameworks to appear and make large leaps to solve problems that have baffled or been too difficult to compute on super computers. At the same time as these crowd sourcing models are made more prevalent consumer based applications will continue to grow in the direction of

completely offloading all computation to a remote machine. Perhaps we shall see more machine hosting companies appear but they will most likely be bought out by large companies like Amazon or Dropbox.

Another perk about distributed application frameworks is the decrease in power consumption for mobile devices. Mobile batteries are becoming more efficient and in parallel DAPF create less consumption of energy on the mobile device. This leads to batteries having much longer life spans.

7. Conclusion

In this paper have used classifications and taxonomy of [1] to identify two types of Distributed Application Process Frameworks (DAPFs) that illustrate examples of Application Partitioning, Thread and Module level parallelism as well as make a relevant interest into modern technologies.

8. References

[1] Shiraz, M.; Gani, A.; Khokhar, R.H.; Buyya, R., "A Review on Distributed Application Processing Frameworks in Smart Mobile Devices for Mobile Cloud Computing," Communications Surveys & Tutorials, IEEE , vol.15, no.3, pp.1294,1313, Third Quarter 2013

[2] Huosheng Hu, Michael Brady, A parallel processing architecture for sensor-based control of intelligent mobile robots. Robotics and Autonomous Systems, Volume 17, Issue 4, June 1996, 235-257.

[3] Calin Cascaval, Seth Fowler, Pablo Montesinos-Ortego, Wayne Piekarski, Mehrdad Reshadi, Behnam Robatmili, Michael Weber, and Vrajesh Bhavsar. 2013. ZOOMM: a parallel web browser engine for multicore mobile devices. In *Proceedings of the 18th ACM SIGPLAN symposium on Principles and practice of parallel programming* (PPoPP '13). ACM, New York, NY, USA, 271-280.

[4] VMware vSphere migration software. http://www.vmware.com/products/vsphere/features-vmotion.

[5] KVM OS Migration information page. http://www.linux-kvm.org/page/Migration

[6] UC Berkeley SETI@home open source software http://setiathome.ssl.berkeley.edu/

[7] Playstation 3 Discover Folding@home http://uk.playstation.com/ps3/support/network/detail/linked224880/item63769/Discover-Foldinghome%E2%84%A2/

[8] Standford University Folding Project http://folding.stanford.edu/