



Many legs make light work.™

The Digipede Framework™ Software Development Kit (SDK)

Table of Contents:

Executive Summary	1
Introduction	2
Architecture.....	3
Solution Scenarios.....	5
Web Applications	5
Transformation.....	6
EDI Applications	7
ETL Applications	8
Digital Media Applications	9
Monte Carlo/Parametric Computing	10
Data Intensive Computing	10
Conclusion	11

©Copyright Digipede Technologies, LLC.

Digipede and the Digipede Network are trademarks of Digipede Technologies, LLC. Microsoft is a registered trademark of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners.

Digipede: Many legs make light work™

Executive Summary

This paper provides an overview of the Digipede Framework SDK, a component of the Digipede Network from Digipede Technologies. The Digipede Network is a distributed computing solution that delivers dramatically improved performance for real-world business applications. Built entirely on the Microsoft .NET platform, it is radically easier to buy, install, learn, and use than other grid computing solutions. With the Digipede Network a company can distribute work across its existing Windows desktops and servers. The Digipede Network is described in more detail in the whitepaper entitled *Distributed Computing with the Digipede Network™*. You can use the Digipede Network to distribute command-line applications and scripts without writing any code, through the user interface called the Digipede Workbench. Add the Digipede Framework SDK, and you can quickly and easily grid-enable your own applications using the Digipede Framework's .NET or COM APIs.

This paper also describes several application scenarios for which the Digipede Framework provides a clean solution. While much attention in the grid community is focused on scientific and technical computing applications, we discuss several mainstream enterprise computing applications for which the Digipede Network can provide order-of-magnitude performance improvements.

Introduction

The Digipede Network (including the Digipede Framework) is built on the Microsoft .NET Framework and takes advantage of Microsoft's Web Services Enhancements (WSE) for improved security. The Digipede Network can be quickly installed and configured on your currently existing network. There is no need to buy more servers, add personnel, increase rack space, or hire expensive consultants. You can set up a grid on your currently existing Windows network in about an hour, and expand it by adding more agents - either on your existing computers, or with new hardware.

Why Digipede?

- Scale out applications and processes for higher performance
 - Distributes application load across Windows desktops, servers, and clusters.
 - Scales from five nodes to thousands
 - Delivers order-of-magnitude increase in speed and throughput
- Increase productivity.
 - Shorter runtimes mean less waiting, more productive work
 - Flexible and powerful APIs (.NET and COM server) enable your developers to grid-enable applications quickly
 - Enable developers to focus on business requirements instead of building an in-house grid or distribution platform
 - Increased use of idle resources raises IT efficiency
- First commercial grid computing solution based entirely on .NET
 - Integrates with Visual Studio .NET for developer productivity
 - Integrates with Windows security for consistency with current practices
 - Uses Web services for ease of implementation
- Rely on a scalable grid computing platform that:
 - Guarantees quality of service
 - Guarantees task completion
 - Integrates data transfer
 - Integrates with your current Windows security
 - Provides job monitoring functions
 - Has been rigorously tested and proven in commercial use
 - Development community with forums and sample code.
- Low total cost of ownership
 - Low introductory price.
 - Radically easier than other grid systems - you can install and administer it yourself

Digipede-enabling your application is quick and easy. Using the Digipede Framework, create a DigipedeClient object to communicate with the Digipede Server, define a JobTemplate, Job, and Tasks, then call SubmitJob(). For many applications, this can be accomplished with as few as 20 lines of code, and the Digipede Network takes care of the rest. With all the demands you face for increased functionality, why spend needless extra time on scalability and performance? The Digipede Network handles all the "plumbing" necessary for secure, reliable, distributed execution of your application, so you can focus on your critical functionality requirements.

Architecture

A basic distributable object is made up of a JobTemplate, a Job and a Task. The JobTemplate defines global information about a Job; for example, which files are required and how to start and monitor a Job. A Job contains more specific details concerning a specific run of a JobTemplate- it contains the definition for one or more Tasks that make up the Job. All defined Tasks can be run in parallel and are independent of each other. Tasks are the work that Digipede Agents pick up from the Digipede Server.¹

Figure 1 shows the relationship among all the elements needed to distribute an application using the Digipede Framework. A Master Application programmatically defines a Digipede Job using the Digipede Framework API. The Job is submitted through the Digipede Framework to the Digipede Server. Each available Digipede Agent checks in with the Digipede Server to find out if there are any tasks that it can run. When the agent finds a compatible Task it obtains that Task from the Digipede Server, downloads any software and files it needs for the distributed application, and executes it.

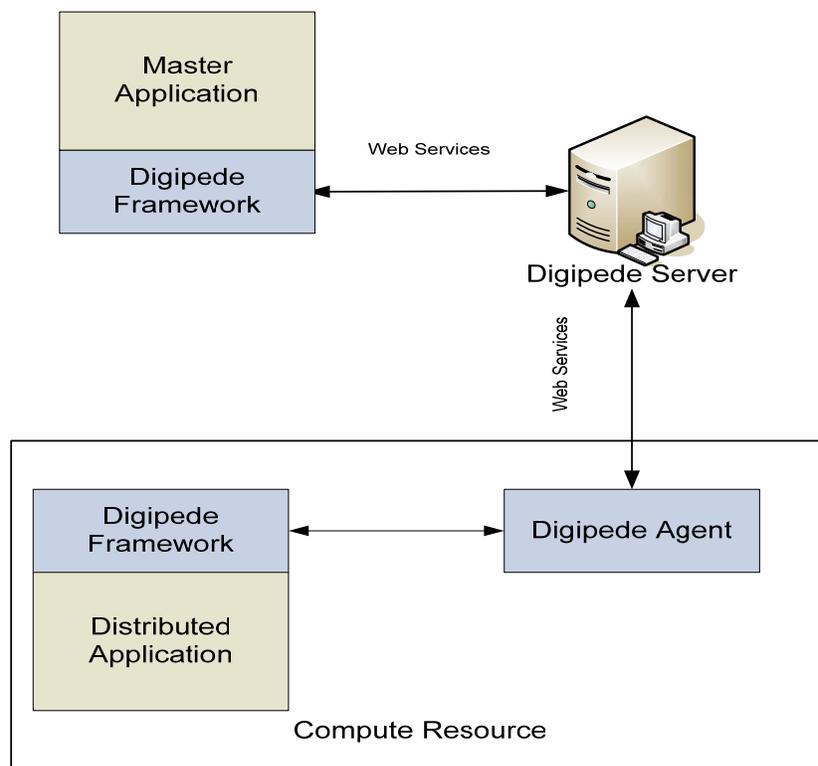


Figure 1. Basic Master Application Architecture

An application that creates Jobs is called a Master Application; the component that is distributed is called the Distributed Application. The Digipede Framework supports several design patterns. A Master Application can start a Job created by the Digipede Workbench, a third party application, or a .NET assembly or COM server. You can build a Master Application using any language or toolkit that supports .NET or COM. This includes Microsoft Visual Tools for Office (VSTO). It is very easy to Digipede-enable a Microsoft Excel spreadsheet by using the VSTO OfficeCodeBehind class. The developer has the choice of using either the .NET or COM APIs.

¹ For more on Jobs and Tasks, please see *Distributed Computing with the Digipede Network™*, and the Digipede Framework SDK Developer's Guide.

Creating a Job for submission to the Digipede Network is very simple:

1. Create a DigipedeClient object. A DigipedeClient object communicates with the Digipede Server and submits jobs. All Master Applications instantiate a DigipedeClient object.
2. Define a JobTemplate. The JobTemplate contains global information about a Job- which files are required and how to start and monitor a Job. The JobTemplate can be preexisting or it can be created on-the-fly. The Digipede Framework can create most JobTemplates automatically. For example, the Framework can create a JobTemplate to distribute a particular .NET assembly simply by passing in a Type from that assembly; the Digipede Network will use reflection to determine which other assemblers are required to instantiate or deserialize those objects on a remote machine.
3. Create a Job. A Job contains more specific details, building on the JobTemplate. Most importantly, a Job contains the Tasks.
4. Create the Tasks and assign them to the Job. A Task defines the work that is to be distributed to an individual compute resource.
5. Use the DigipedeClient object to submit the Job to the Digipede Server.

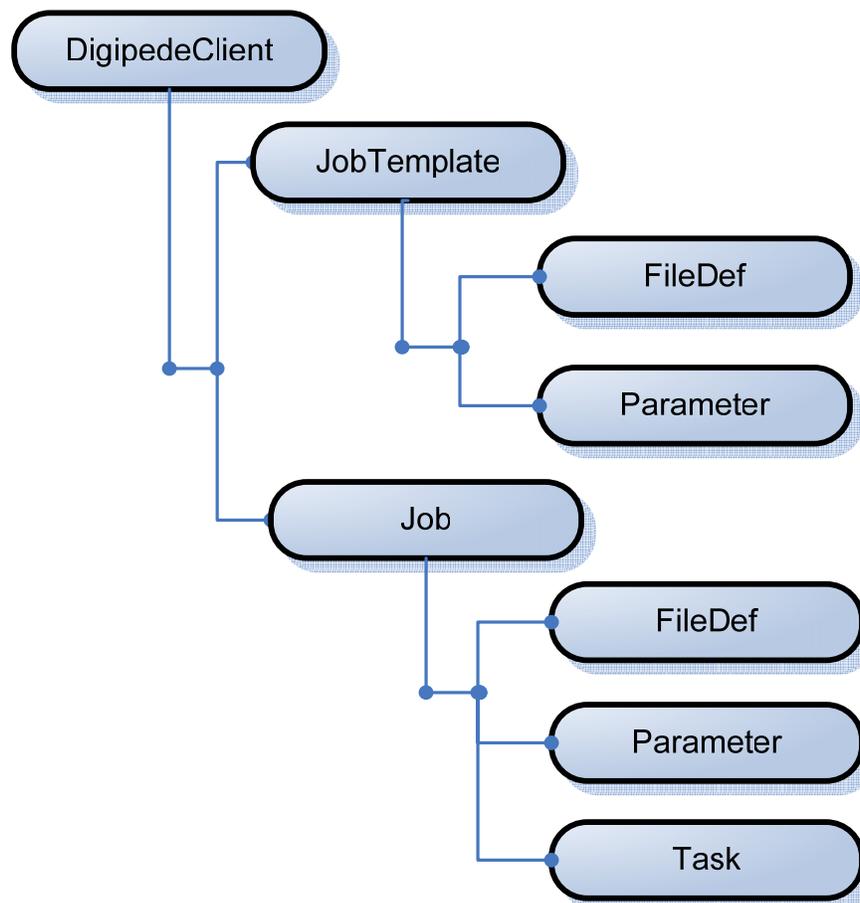


Figure 2. Sample Digipede Object Relationship

Solution Scenarios

The Digipede Framework can be used to solve a variety of application performance problems. The solution scenarios that follow barely hint at the breadth of problems that can benefit through reducing processing time or scaling-out the application.² There are multiple ways to solve each of the scenarios; Digipede lets you decide the technique that works best for you. Many scenarios can be addressed simply by using the Digipede Workbench; when you need more control or custom code, the Digipede Framework is both powerful and flexible.

Web Applications

Figure 3 shows a simple web application, in which the users access a web page that extracts data from a database, converts the data to a file, then serves the file back to the requesting user. The architectural diagram in Figure 3 represents the typical application design.

In this scenario the computationally intensive conversion process is run on the web server, and a single run is not a strain on the server. The strain occurs when multiple, simultaneous requests happen. In this architecture, a single machine is handling all of the processing. When multiple requests come in at once, the machine can either handle them serially (in which case some users experience much longer delays than others) or, if the software permits, handle them in parallel (in which case the CPU becomes overworked, and every user experiences unacceptable delays).

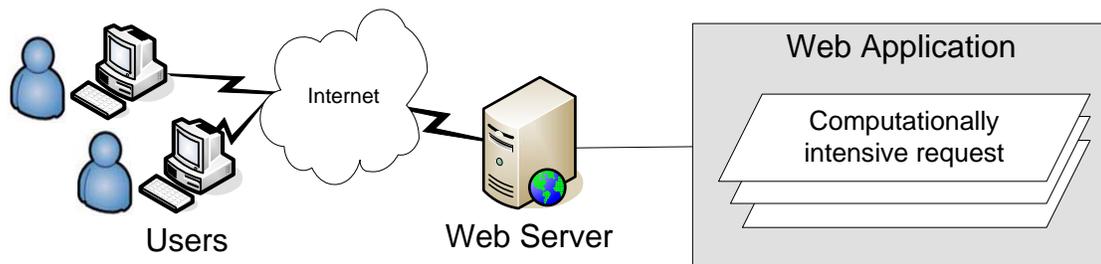


Figure 3. A Basic Web Application Architecture before Digipede

Using the Digipede Network with the Digipede Framework the web developer can quickly and easily move the CPU-intensive conversion process off the web server and distribute it to other machines on his or her network. Figure 4 shows the new architecture for the web application. For each request the computationally-intensive file conversion is distributed to the compute resources. This allows many requests to be processed simultaneously without adversely affecting performance, and also frees the web server up to respond promptly to users. The distributed code can be cached on each compute resource so that only the request data is serialized across the network, keeping bandwidth utilization to a minimum.

² Additional solution scenarios are posted on the Digipede Community Forum, which is open to all Digipede customers.

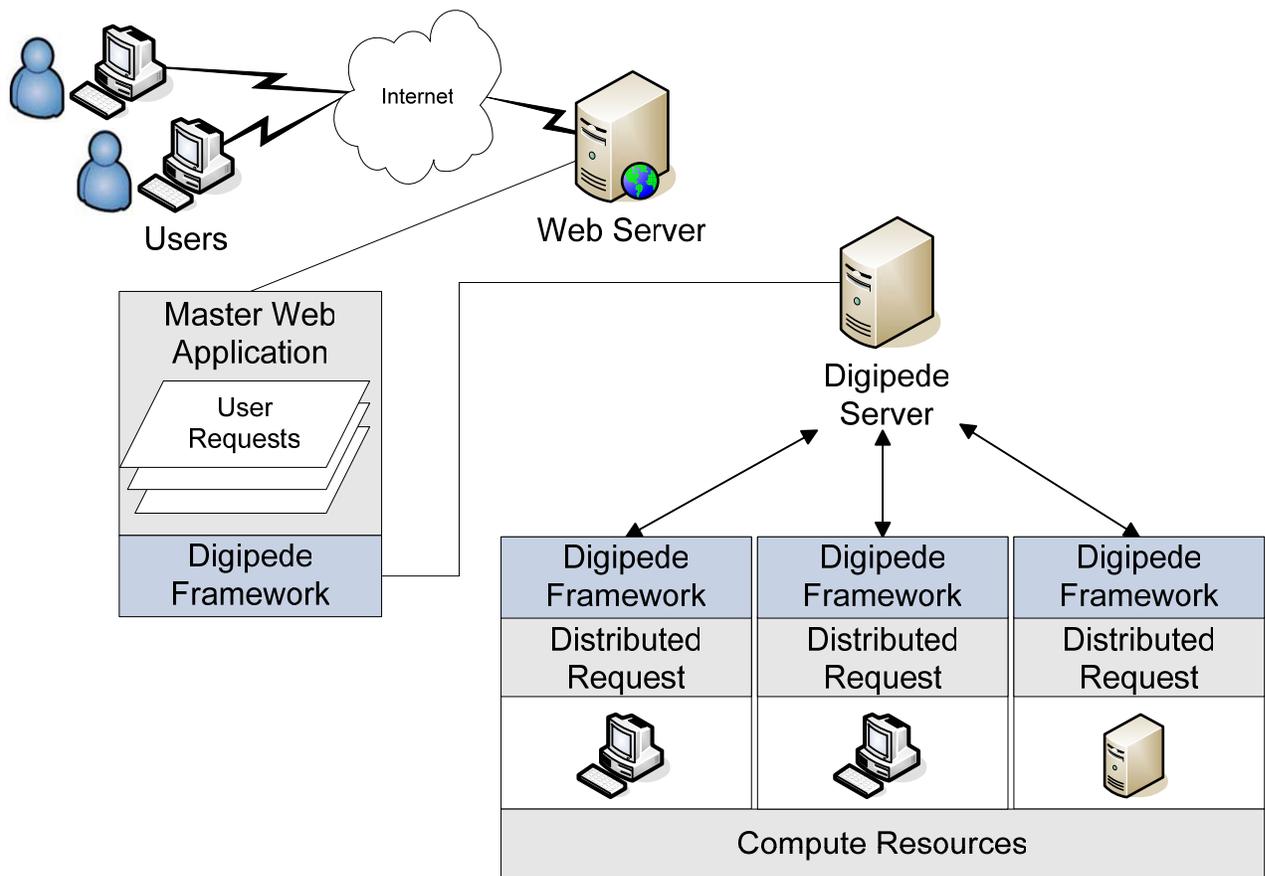


Figure 4. A Basic Web Application Architecture after Digipede

The web developer has just scaled out the file conversion process across the network. By using available compute resources the web developer has avoided the costs of scaling up, rolled the solution out quickly, and placed the burden of distribution firmly on a system specifically designed and tested for that purpose.

Transformation

Transformation of files and data is very common:

- EDI transactions from one company's computer systems are sent to another company. The companies often use different technologies or processes that require that the incoming transactions undergo a transformation before they can be loaded into the receiving company's systems.
- Data collection from sensors and devices is often stored in a proprietary file format specific to the collection hardware. To clean and move the data into a searchable form (i.e. a database) the company will have to transform and normalize the raw data.
- Digital media file transformations such as 3-D image rendering is very computationally intensive because the rendering applications must orient the wire frame, apply texture, apply special effects, adjust lighting, and finally render the 3-D image into a 2-D image.

CPU intensive and time demanding transformations can often run in parallel. The benefit of parallelizing these activities is that the end-users will get the results faster. This increases productivity and saves money.

EDI Applications

EDI transaction mapping for many companies happens daily. In some instances the transactions come one at a time but in a flurry, at other times as one large batch delivery. When the transactions are batched the sequential mappings can take a long time to run. When they come in a flurry the mappings can strain the network. Often IT departments purchase dedicated machines to support these peak through-put times; resulting in a large capital investment that is often idle.

Figure 5 shows a sample bulk EDI mapping. In this scenario a company receives a large EDI transaction containing customer Medicare updates. The information includes customer coverage status, including changes to coverage, for thousands of customers; delays in updating this information in the company's database can lead to improper reimbursements and increased customer complaints.

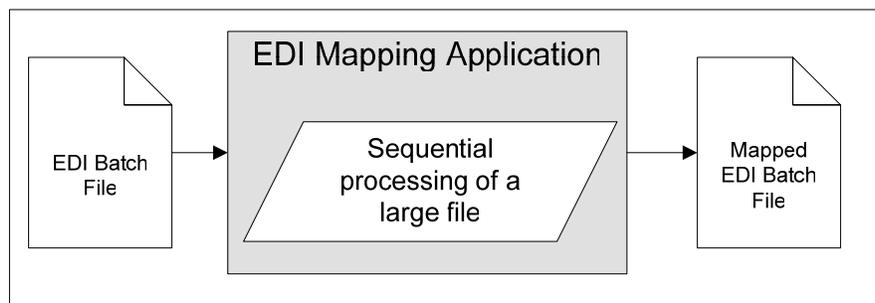


Figure 5. A Basic EDI Batch Application Architecture before Digipede

One technique for faster processing of large batch EDI transaction files is to segment the large file into smaller ones and distribute the mapping transformation across the network. Once all the segmented files have been mapped, the developer can rebuild a new mapped batch file and perform the standard load, consequently adding parallelism to the time-intensive part of the EDI process.

Figure 6 shows the new application architecture. The company receives a large EDI batch file from their trading partner. The EDI Master Application opens the batch file and segments it into smaller units. These smaller units are assigned to Digipede Tasks and distributed to the compute resources where the mapping transformations occur. The mapped results are then used to create a new mapped EDI batch file which is then handed off to the loading software.

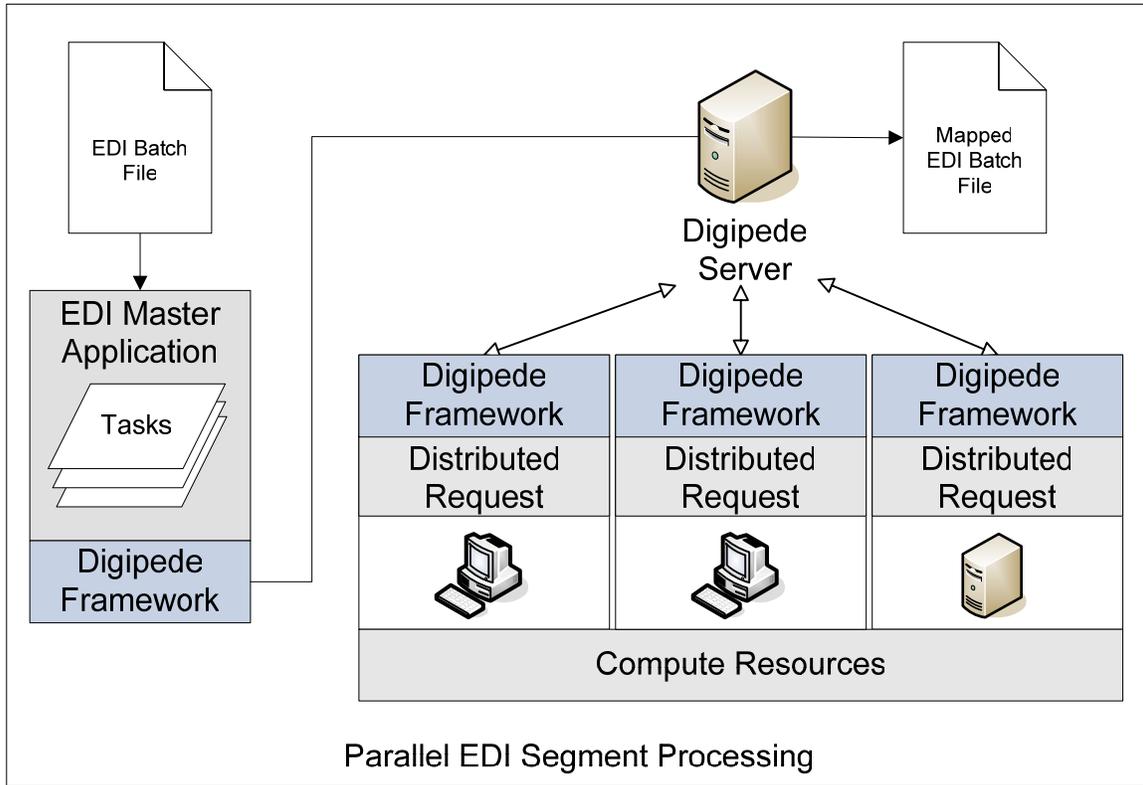


Figure 6. A Basic EDI Batch Application Architecture after Digipede

By using the Digipede Framework and currently existing compute resources, the developer has reduced the processing time required to map a large EDI batch file.

ETL Applications

Figure 7 shows a sample ETL application. In this scenario a company is collecting mobile phone data by utilizing mobile equipment (in a specially equipped van) that makes calls and collects data for each carrier. Each phone connects to a different cellular provider and the equipment tracks the quality of the connection. The equipment creates a data file for each provider for each hour of collection. An eight hour drive, with four carriers, produces 32 files. Each file must then be converted from the proprietary collection file format to a standard data load format. Without distributed execution each of the 32 files must be converted serially. Any data cleansing algorithms must also run serially.

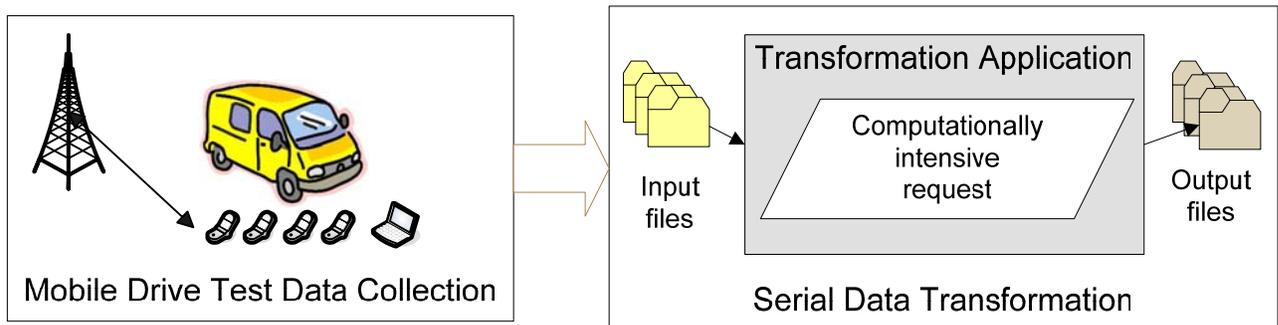


Figure 7. A Basic ETL Application Architecture before Digipede

Optimizing this process will save money, increase productivity, and positively affect revenue. Since the data files are independent of each other, the developer can run the transformation processing in parallel, and, once a file transformation is completed, start normalizing the file.

Figure 8 shows the new application architecture. Each file transformation is sent to a compute resource as a Task and executed in parallel with the others. This allows multiple simultaneous file transformations, which significantly reduces the file transformation times. The developer can build workflow into the master application so that when a Task (file transformation) completes, the master application immediately creates a Job that initiates cleansing of the generated file.

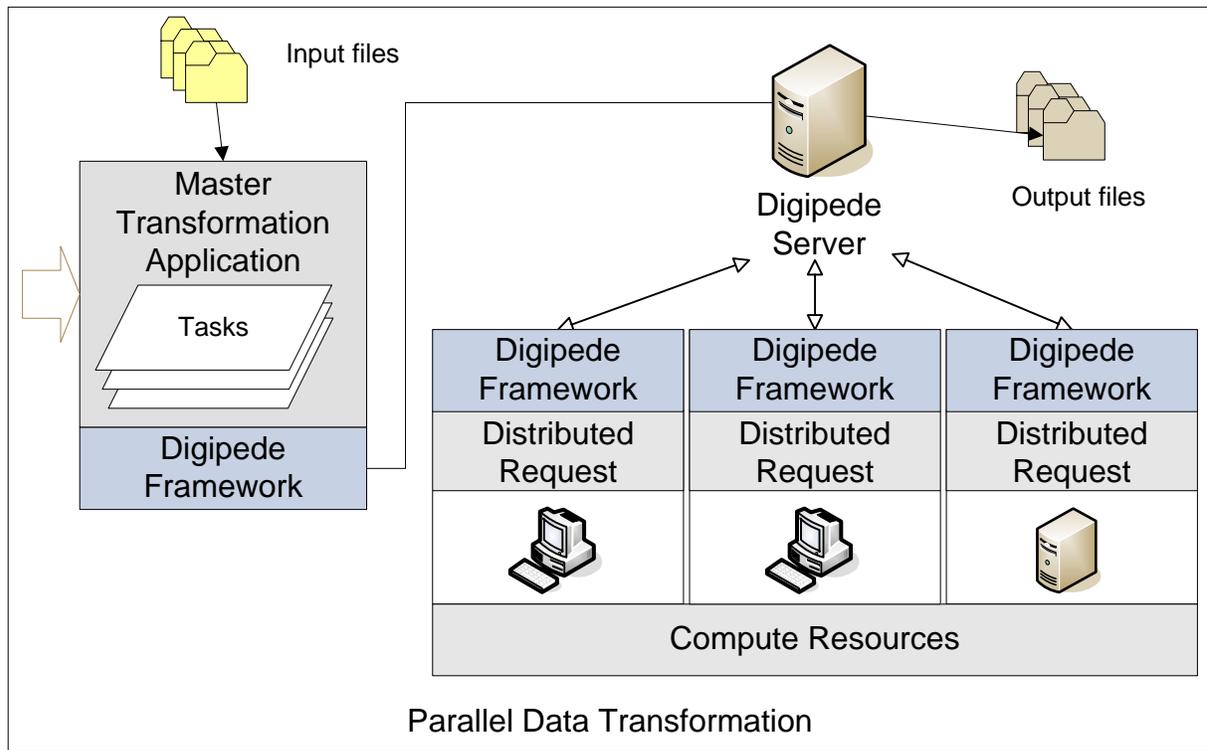


Figure 8. A Basic ETL Application Architecture after Digipede

The developer has just distributed a file transformation process across the network. By using the Digipede Framework and currently existing compute resources, the developer has significantly reduced the time required to extract and transform a drive data set. This allows the company to process to the new data more quickly.

Digital Media Applications

Digital media encompasses many different types of file formats and processes. One thing they have in common is that the best file format for creation and editing is not necessarily the best file format for presentation. This means that there must be one or more file transformations and these transformations are almost always computationally intensive.

Image rendering was one of the earliest uses for grid computing. One scenario is that of a graphic artist who has prepared a 3D model for animation and is ready to render the model to a file for presentation. Taken one frame at a time the rendering process will take several hours. The rendering process can be accelerated by breaking the input file into frames and distributing the

frames to different compute resources. Using the Digipede Network to distribute the files and start the render application, the artist can utilize many machines simultaneously and dramatically reduce the rendering time.

Monte Carlo/Parametric Computing

Parametric and Monte Carlo algorithms are great examples of processing that is easy to distribute. These algorithms are run many times to generate data points that simulate real-world conditions. Distributing these computations across the network allows faster completion of the calculations and enables running finer simulations. This translates into increased productivity and accuracy for the person running the simulation.

Microsoft Excel

The Digipede Framework works well with Microsoft Visual Tools for Office (VSTO). By using the OfficeCodeBehind class defined by VSTO and .NET, you can easily set up the Digipede Framework calls to initiate parallel processes such as Monte Carlo algorithms. The Digipede Framework can also be used with older versions of Excel that require a purely COM solution. Many currently existing Excel applications consist of a master Excel spreadsheet that controls worker spreadsheets. The Digipede Network can be used to distribute the computation of the worker spreadsheets, using only VBA and the Digipede Framework COM interfaces.

For example, a risk analyst uses a Monte Carlo simulation to analyze the risk of a new insurance product. The analyst runs thousands of simulations to generate enough data points to produce a realistic simulation. Generating each data point sequentially takes hours. By using the Digipede Framework, the analyst changes the simulation so that each data point is generated at a different compute resource. This allows the analyst to run the same number of trials in a much shorter period of time.

Scientific Computing

There are many computationally intensive problems in science, such as weather forecasting, wind shear studies, and PET scan imaging. Some of the computations are used in modeling and simulations; others in real-time analysis and data conversion. Many of them can run in parallel, resulting in faster runs.

Data Intensive Computing

Many large data sets are not well suited for storage in relational databases. Often these data sets are stored in flat files and can be broken up among multiple compute resources. The Digipede Network provides a tool to easily search and mine those data sets.

Search

BLAST is a program used widely in bioinformatics to search protein and nucleotide databases. In order to search a huge data set quickly (for example, the human genome is more than 3 gigabytes), the data set is broken into segments. Any given query is searched against each segment of the data set; those searches can be run in parallel. By segmenting the data sets across a network and distributing the search query, a researcher can get results back much faster. This technique is also used by Microsoft, Google, and others to drive their powerful search engines. By using the Digipede Framework, this search technique becomes quickly available to researchers. Running BLAST on a network of Windows desktops, servers, and cluster nodes can increase the throughput of this critical analysis tool by two or even three orders of magnitude. The researchers can focus on their analysis without having to worry about building the supporting infrastructure components.

Conclusion

There are many applications and industries that can benefit from distributed, parallel, and grid computing. Improved performance, scaled out solutions and a robust distribution platform are all available through utilization of the Digipede Network. With the choice of .NET and COM APIs in the Digipede Framework, developers can continue to use the tools and technologies with which they are already comfortable.