Int. J. Bioinformatics Research and Applications, Vol. X, No. Y, XXXX

Scaling up genome annotation using MAKER and work queue

Andrew Thrasher

Department of Computer Science and Engineering,
University of Notre Dame,
Notre Dame, IN, USA
Email: athrash1@alumni.nd.edu

Zachary Musgrave

Yelp, Inc.,
San Francisco, CA, USA
Email: ztm@yelp.com

Brian Kachmarck, Douglas Thain and Scott Emrich*

Department of Computer Science and Engineering,
University of Notre Dame,
Notre Dame, IN, USA
Email: bkachmar@nd.edu
Email: dthain@nd.edu
Email: semrich@nd.edu
*Corresponding author

Abstract: Next generation sequencing technologies have enabled sequencing many genomes. Because of the overall increasing demand and the inherent parallelism available in many required analyses, these bioinformatics applications should ideally run on clusters, clouds and/or grids. We present a modified annotation framework that achieves a speed-up of 45x using 50 workers using a *Caenorhabditis japonica* test case. We also evaluate these modifications within the Amazon EC2 cloud framework. The underlying genome annotation (MAKER) is parallelised as an MPI application. Our framework enables it to now run without MPI while utilising a wide variety of distributed computing resources. This parallel framework also allows easy explicit data transfer, which helps overcome a major limitation of bioinformatics tools that often rely on shared file systems. Combined, our proposed framework can be used, even during early stages of development, to easily run sequence analysis tools on clusters, grids and clouds.

Keywords: genome annotation; work queue; distributed computing.

Reference to this paper should be made as follows: Thrasher, A., Musgrave, Z., Kachmarck, B., Thain, D. and Emrich, S. (XXXX) 'Scaling up genome annotation using MAKER and work queue', *Int. J. Bioinformatics Research and Applications*, Vol. X, No. Y, pp.xxx–xxx.

Biographical notes: Andrew Thrasher received a BA in Computer Science, Mathematics and Physics from Anderson University and the MS and PhD in Computer Science and Engineering from the University of Notre Dame. He is currently a Senior Software Engineer in Computational Biology at St. Jude Children's Research Hospital.

Zachary Musgrave is a software engineer on the Infrastructure team at Yelp, Inc. He earned the MS in Computer Science and Engineering from the University of Michigan at Ann Arbor; previously, he earned the BS in Computer Science and the BA in English Writing from Clemson University in South Carolina. He now contributes to open-source projects involving cloud management and map/reduce style data processing.

Brian Kachmarck received a BS in Computer Science and Engineering from the University of Notre Dame. He is now a Software Development Engineer at Amazon (Seattle).

Douglas Thain is an Associate Professor of Computer Science and Engineering at the University of Notre Dame. He received the PhD in Computer Sciences from the University of Wisconsin and the BS in Physics from the University of Minnesota. His research is focused on the design of large-scale computing systems for science and engineering.

Scott Emrich is an Assistant Professor of Computer Science and Engineering at the University of Notre Dame. He received the PhD in Bioinformatics and Computational Biology from Iowa State University and a BS in Biology and Computer Science from Loyola College (Maryland). His research is focused on genome informatics with an emphasis on arthropod research.

This paper is a revised and expanded version of a paper entitled 'Shifting the bioinformatics computing paradigm: a case study in parallelizing genome annotation using MAKER and Work Queue' presented at the '2nd IEEE International Conference on Computational Advances in Bio and Medical Sciences (ICCABS 2012)', Las Vegas, NV, USA, 23–25 February 2012.

1 Background

Bioinformatics has emerged over the past decade as a major producer of digital data and consequently a large consumer of computing resources. The tremendous explosion in sequencing capability with greatly reduced costs has enabled scientists to sequence non-model organisms. Moreover, they are also (re)sequencing larger numbers of genomes from a population including humans (Durbin et al., 2010). To help facilitate their scientific investigations, many genome projects also perform additional sequencing such as Expressed Sequence Tags (ESTs) and RNAseq. All these factors contribute to the large quantity and variety of data available in a modern genome project.

Parallel to the increase in sequencing power, algorithms for genome analysis and assembly have improved. Annotation, however, remains a bottleneck given it historically has been a highly human/curator intensive process. As the number of genomes sequenced outpaces available curation resources, there have been ongoing attempts to automate and provide reasonable annotations from mostly computational approaches. These methods, however, are often highly computationally intensive because they integrate a variety

of data sources to increase annotation quality. Many of these annotation tasks have natural inherent parallelism. We previously demonstrated that a distributed computing framework can allow developers to leverage a variety of heterogeneous computational resources in a lightweight and relatively easy to program manner (Lanc et al., 2011; Moretti et al., 2009; Bui et al., 2011; Thrasher et al., 2010). There are a variety of similar existing frameworks. A common feature of a good framework is an easy to utilise API, and the main contribution of this work is a case study that utilises one to provision a diverse collection of computing resources, perform for data-intensive annotation using MAKER (Cantarel et al., 2008), and their combination into open source distributed bioinformatics tools for the research community. Additionally, we provide an evaluation of the performance of our MAKER modifications using the *Caenorhabditis japonica* genome with supporting evidence totalling 180 megabases and achieve a speed-up of 45x using 50 workers.

We also show that this framework is capable of harnessing the power of cloud computing frameworks such as Amazon's EC2. Our evaluation is done on the *Anopheles gambiae* PEST genome of 278 megabases with 167 megabases of supporting evidence. (See Table 1 for summary of data set sizes) Gene finding in novel genomes can follow an evidence-based or *ab initio* framework. In the former approach, well-characterised protein data from related organisms along with expression data (including newer generation sequences) are used to build gene models based on these experimentally observed intron-exon boundaries. Because these data are expensive to generate and do not cover everything, *ab initio* approaches are used to predict intro-exon boundaries using only a sequence of a new genome. Popular *ab initio* approaches include GLIMMER (Delcher et al., 1999), FGENESH (Salamov and Solovyev, 2000), GenScan (Burge and Karlin, 1997), SNAP (Korf, 2013) and Augustus (Stanke and Waack, 2003).

Table 1 MAKER data sets for various organisms

Organism	*Genome*	*EST*	*RNAseq*
Anopheles stevensi	207 MB	2.2 MB	1893 MB
Glossina morsitans	366 MB	48 MB	0 MB
Rhodnius	699 MB	6 MB	372 MB

Each *ab initio* approach uses complex probabilistic models trained on either a small subset of known genes or a subset of likely genes. For example, GLIMMer uses long open reading frames found in bacterial genomes for training because they are unlikely to occur by chance (Delcher et al., 1999). The commercial gene finder FGENESH relied on high confidence EST alignments for training has been reported to work well for plant genomes such as maize (Emrich et al., 2007).

Here, we use MAKER (Cantarel et al., 2008) that incorporates both Augustus and SNAP into the gene prediction pipeline. Both Augustus and SNAP rely on Hidden Markov Models (HMMs) as their underlying probabilistic model. The key innovation in Augustus is modelling of intron lengths on top of the HMMs, which can improve results (Stanke and Waack, 2003). SNAP uses similar intron length modelling that is very similar to GenScan (Burge and Karlin, 1997). The most important SNAP features for large-scale gene prediction in novel genomes are flexible state diagrams to allow

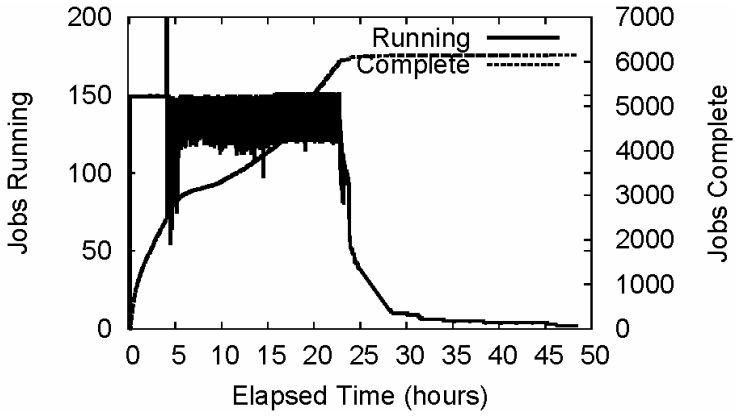
changing the underlying HMM and flexible inputs such as the order or Markov model and weight matrix used. This allows for both customisation and 'boostrapping' in which models from related species iteratively improve SNAP training sets (Korf, 2013).

One additional component of the SNAP/MAKER pipeline is use of the Core Eukaryotic Gene Mapping Approach (CEGMA; Parra et al., 2007). CEGMA, which is provided by the author of SNAP, contains a set of 458 proposed core genes that are present in many species. Specifically, these highly conserved genes provide a highly reliable set of training models because their conservation makes it easier to derive exon-intron structure (Parra et al., 2007). This method utilises profile-HMMs to produce gene structures.

2 Results and discussion

To scale up to a much larger problem, we used our modifications to annotate a 207 Mb mosquito genome with 2 Mb of EST and 1893 Mb of RNAseq supporting evidence. This required 2911 min with a similar Condor worker pool of 150 nodes. The computation was able to use the full 150 nodes for 1363 min, but then a long-tail effect comes as shown in Figure 1.

Figure 1 A run of the distributed maker platform using a 207 Mb genome with additional supporting data



We have performed additional work to enhance our WQ-MAKER's usefulness for the community at large: we added our tool to the popular Galaxy framework (Giardine et al., 2005; Goecks et al., 2006; Blankenberg et al., 2010) for bioinformatics. The tool runs our work queue version of MAKER by default, but could be made to run either the serial version of MAKER or the MPI version of MAKER by changing the executable that gets called. As MAKER relies on a shared file system for distribution of work, galaxy needs to be installed on a shared file system or at least have its job working directory on a shared file system for the work queue or MPI versions to function properly. We are working to eliminate this restriction in future versions of WQ-MAKER.

2.1 Amazon EC2

2.1.1 Introduction

Commercial cloud computing platforms are an attractive alternative for those who want to use WQ-MAKER but have limited access to resources. Using WQ-MAKER in the cloud provides the ability to provision instances on-demand while paying only for what is used. Here we explore the extensibility of WQ-MAKER to the cloud and specifically Amazon EC2, which is a leader in the commercial cloud computing market for bioinformatics applications. We ran WQ-MAKER in EC2 using the *A. gambiae* PEST genome assembly (8982 sequences, 278 MB) from Vectorbase (Lawson et al., 2009), assembled transcripts (14974 sequences, 26 MB), unassembled EST reads (21729\7 sequences, 134 MB), and all available protein sequences (14,324 sequences, 7.5 MB). All data are available from the website www.vectorbase.org

2.1.2 Set-up

To run WQ-MAKER in EC2, some initial set-up work must be performed. Amazon offers a range of instance types, each with a different amount of computing capacity and different cost per instance-hour.

All instances must have an *image* attached. The Amazon Machine Image (AMI) contains the operating system and application software for the virtual machine. A pre-defined AMI with Fedora 12 operating system was selected for these runs. Once the instance was created, we installed MAKER as instructed by the developers and all dependencies of the software on the virtual machine. Because WQ-MAKER requires a shared file system, we also installed NFS on the virtual machine to meet this requirement (Linux NFS, see http://linux-nfs.org). Every instance created for the WQ-MAKER personal cloud will need the same software, so this image was saved as a custom AMI and reused by each instance we created. For interested users, this AMI has been made public and the server AMI is *ami-8dab01e4* and the client AMI is *ami-f9ab0190*.

Amazon provides Elastic Block Store (EBS) as volumes. We created an EBS volume in Amazon, uploaded all the input *Anopheles* data files to the EBS, and saved the EBS for future use. We mounted the EBS volume to the master instance of our WQ-MAKER cloud for all experiments below.

2.1.3 Selecting an instance type

Amazon provides a variety of instance types to select for virtual machines. For example, the medium standard on-demand instance provides 1 virtual core, 3.75 GB of memory and costs $0.160 per hour. The extra large high-CPU on-demand instance provides eight virtual cores, 7 GB of memory, and costs $0.660 per hour. It is important to select an instance type that maximises computing capacity while minimising the cost for a WQ-MAKER job. Because a single core can be used for one work queue worker, we chose to use the extra large high-CPU on-demand instance to minimise total instances in the cloud. This also minimises the total cost for the job (data not shown) for this application.

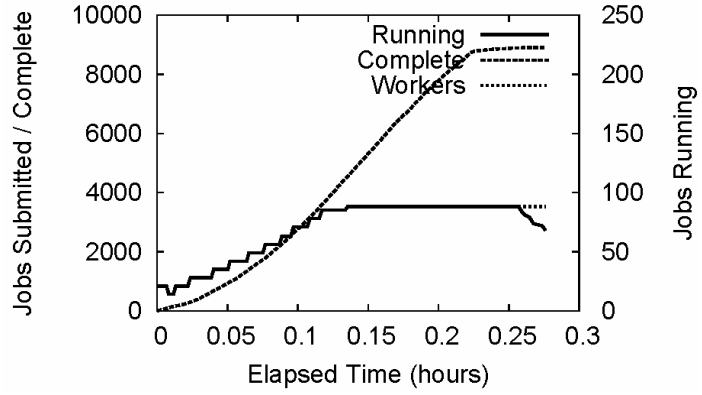
2.1.4 Running WQ-MAKER in EC2

Master processes can be run inside or outside of the cloud. For this study, we ran WQ-MAKER entirely in the cloud via a virtual machine with a custom AMI to act as the master for our WQ-MAKER application and a variable number of worker instances. NFS in the master virtual machine must give access to each worker, so we added the worker machines' hostnames to the file */etc/exports*. Each worker was given read and write permissions as well as the permission '*no_root_squash*' that allows the worker to access NFS as root and can add or edit the necessary files to the WQ-MAKER directory. On the pre-defined AMI provided with this work, the script '*add_nfs_client.sh*' is available to add a worker to the master NFS server, '*start_nfs_server.sh*' to start the server and '*start_nfs_client.sh*' is available to connect the client to the server. In this pilot we started the work queue worker application manually with a pointer to the master virtual machine; however, this only has to be done once at the start of the WQ-MAKER application.

2.1.5 WQ-MAKER in EC2 results

We ran the *A. gambiae* PEST genome through the WQ-MAKER pipeline in Amazon's EC2 with approximately 100 workers (13 extra-large EC2 instances). Using this configuration required less than 20 min to complete computation (see Figure 2). Because an EC2 user must pay for a full hour of computation for each instance, the total cost for this run was $8.32 (13 machines at $0.640 per hour). This is a fairly low cost to pay for the annotation of a full genome given the cost of even one dedicated multi-core research server. We conclude that MAKER can be used both on private and commercial clouds using our work queue approach.

Figure 2 The distributed maker platform was run in Amazon's EC2 using a using a 278 Mb genome with additional supporting data



2.2 Recommendations

It is well-established given the increasing popularity of cloud-based solutions many bioinformatics applications have highly parallel components. Based on our experience developers should consider changes to several key areas of the current bioinformatics computing paradigm.

First, many bioinformatics applications rely on a shared file system when explicit transfers may help run on more heterogeneous resources. This also helps with debugging, both for the developer and the user, by allowing them to log and identify problems with the application that are caused by network failure and other issues which have prevented a worker node from accessing key files. Explicit transfer must be tightly controlled in environments like Amazon where transfer into and out of a cloud incurs cost but running a master process in the cloud overcomes this concern. More importantly, a framework with build in fault tolerance like work queue or make flow alleviates the need for file-based check pointing, which often limits the scalability of the application regardless of the amount of parallelism inherently available.

Second, common bioinformatics operations are often abstracted into libraries such as BioPerl. While convenient for developers and users, in more heterogeneous environments like ad hoc clouds the library must be available across all compute nodes by either being installed on each node or on a shared file system. Neither of these are desirable situations for large-scale commodity-driven computation. We have found a paradigm similar to the functional programming revived in Google's Map-Reduce (Dean and Ghemawat, 2008) is preferred as compiled and statically linked C executables can be more easily transferred around to multiple systems in our experience compared with Perl or Python.

Third, a distributed computing framework such as work queue enables the utilisation of an application across a variety of execution environments (e.g. an SGE cluster, Amazon EC2). In fact, we have used this framework effectively to harness Condor, SGE, individual workstations, and EC2 compute cycles for large bioinformatics workflows like MAKER. The clear advantage of a system such as this is that it allows the user to utilise any and all computing resources that are currently available to them (or can be purchased from a third party) offering elastic scaling for these applications without refactoring the underlying code.

3 Conclusions

We present a modified MAKER annotation tool capable of being run on a variety of distributed computing resources and commodity resources. This modification will enable even small labs to leverage additional computing power to complete the annotation of their genomes. Additionally, this technique is generally applicable across bioinformatics tools. Tool developers can use the work queue platform to enable their application to be run across a variety of computational resources without having to know the details of all of the possible distributed environments.

The use of a framework for distributing bioinformatics computational problems across distributed computing resources needs to become commonplace. The framework must also be portable across a variety of systems, not tied to one particular system. Additionally this implies that tool developers need to consider the data transfer of their applications. Rather than simply relying on the implicit data transfer of a shared file system, tool developers should utilise frameworks that have explicit data transfer syntax. This will allow portable applications that can be of use to people in a variety of environments without being restricted to a particular implementation. Further, we believe tool developers need to be conscientious in their library selection. Utilising libraries such

as BioPerl inside of an application speeds implementation but greatly limits scaling in a heterogeneous distributed computing environment. Having these libraries utilised within an application will require them to be available on the computation nodes which makes distributed computing efforts much more difficult.

4 Methods

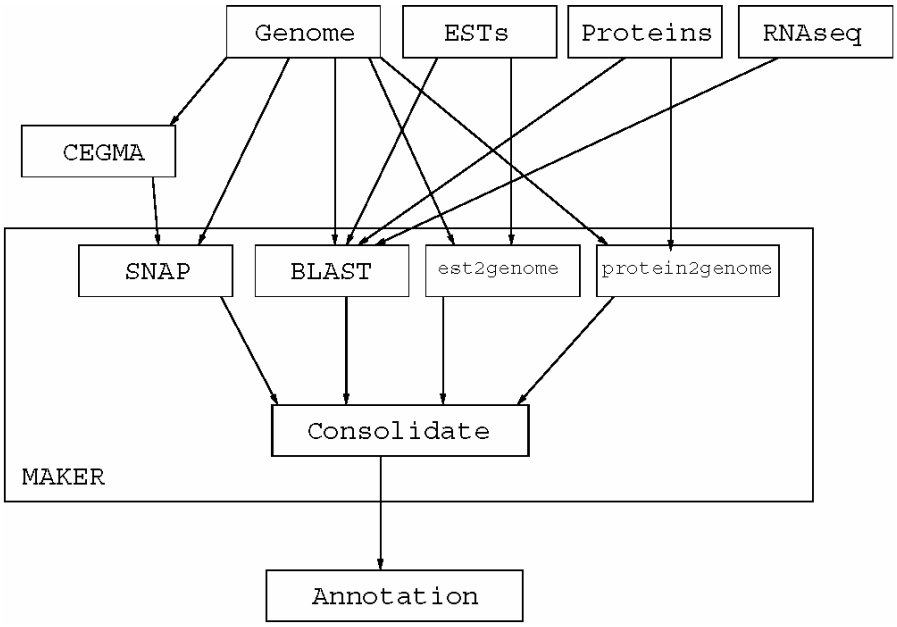
MAKER is a genome annotation utility produced at the University of Utah. It is a pipeline for reconciling the output of many different annotation tools. To do this, it first runs independent tools on the input data, usually a genome and supporting information such as proteins and ESTs. It then combines the output of these tools and produces a consensus annotation for the genomic regions. The standard MAKER instance is configured to run in either serial mode or in parallel using MPI (Foster, 1995) while relying on a network file system. It is specifically aimed at smaller genome projects. The authors describe the accumulation of unannotated genomes that continues to occur as sequencing costs drop and assembly algorithms improve. They assume that the annotation step has become the bottleneck in the genomics project pipeline.

MAKER operates on data in 'tiers'. Each tier is composed of FASTA-formatted nucleotide sequences with potential comparisons to supporting data such as proteins, ESTs, RNAseq, and other relevant information. Tiers also include parameters specified by the user for use with the various tools that provide output to MAKER such as Basic Local Alignment Search Tool (BLAST; Altschul et al., 1990), the ab-initio gene predictor SNAP (Korf, 2013), exonerate (Slater and Birney, 2005), and cross_match from the Phrap suite (Green, 1994–1999). These tools are often tied together using the BioPerl (Stajich et al., 2002) library, which allows users with a basic knowledge of a scripting language to complete complex tasks using functions provided in this library. This also allows users to perform common tasks, such as parsing BLAST output. This variety of applications and libraries makes parallelisation a complicated task, which we discuss here.

Tiers are computed in order. If a tier fails to complete, MAKER re-runs the tier by default up to a specified threshold of attempts. Each tier, composed of pieces of the input data, utilises the various tools of the pipeline. MAKER then takes the output of each tier and consolidates them into a final annotation (see Figure 3).

MAKER tracks the progress of the annotation using a log file. This file is maintained in a shared file system and, in the case of MPI MAKER, each worker node attempts to gain a lock of the file and then write to the file what the worker is currently doing. This allows the 'master' process to track the current state of the annotation and also to restart it in the event of failure or termination. This set-up, however, makes distributed computing difficult as workers from a cluster, grid or cloud must wait to obtain the lock of the file before it can begin. If it fails to obtain the lock for a certain length of time, the worker has to fail. Recovery of this worker depends on the nature of the underlying distributed framework but may add latency in systems such as Condor where jobs do not start instantly.

Figure 3 MAKER overview. MAKER uses a number of external executables to produce an annotated genome from various evidence



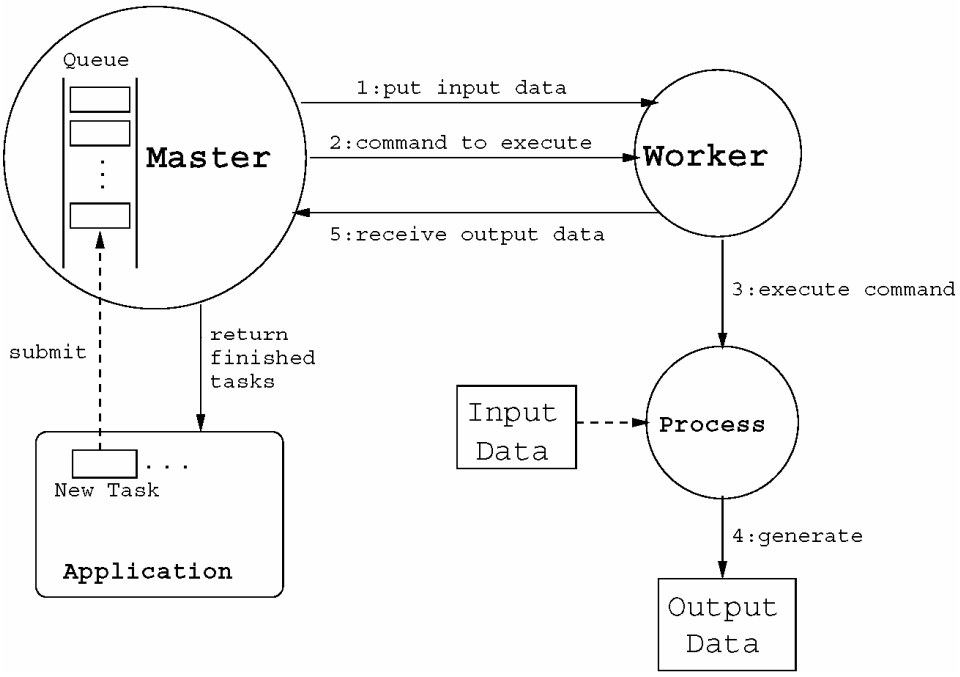
Even so, the tiered nature of MAKER is a prime candidate for using distributed resources. Many of the annotation tools are standalone executables and require a couple of input data chunks. If there was explicit data management the application would be easily portable and practical in various heterogeneous distributed computing environments. By relying on a network file system for implicit data transfer and management, on the other hand, the user base is locked into distributed computing resources with attached networked file systems. This makes utilisation of popular resources like Amazon's EC2 platform very difficult. Note the use of a common log file on a networked file system compounds this difficulty. Ideally, the master process should be capable of tracking tasks that it has dispatched and of determining success or failure from worker responses. This would also eliminate the bottleneck of multiple workers trying to synchronously access the log file and as the problem scales up to tens and hundreds of workers, which could result in many being forced to wait.

In the following section, we provide a description of work we have undertaken to address some potential limitations of the current MAKER implementation that make it difficult to utilise a distributed computing environment. We provide this description to show the relative simplicity of using a distributed computing framework to assist an application in being useful across a wide variety of execution environments. The resulting tool is available to the broad scientific community and has been integrated into a popular workflow (Galaxy) to maximise its application.

In addition to the communities targeted by the authors of MAKER, there are many others with growing genome annotation needs and constraints on resources. There are large genome project initiatives such as the 1000 genomes project (Durbin et al., 2010)

and the Anophelene cluster project (Besansky, 2008). These genome projects focus on comparative genomics, but may still require annotation services based on the large amounts of supporting data and related annotations generated, in which MAKER is well-suited to handle. The bottleneck is in the implementation. Users of MAKER are confined to either serial mode or MPI mode on a dedicated cluster. Many users and institutions have access to a variety of resources such as Condor (Thain et al., 2002), LSF (LSF Home Page, see http://www.platform.com) and SGE (Gentzsch, 2001). We believe bioinformatics applications should be capable of utilising diverse systems such as clusters, clouds and grids to maximise availability to the biological user community.

Figure 4 The general framework of a work queue application. An application creates a queue using the work queue API and submits tasks to the queue specifying the data to be processed, the executable and a command string. The queue then sends the information to a worker that is connected for processing. The worker returns the output to the queue that either marks the task completed and returns the output or marks the task failed and resubmits to another worker



To accomplish this, we have modified the MAKER toolset to utilise a variety of distributed resources. We decided to rely on our prior knowledge and work with the work queue framework (Bui et al., 2011). Work queue is a lightweight master/worker implementation that uses worker processes which are submitted to a batch system. This work queue framework is available as part of the CCTOOLS package (http://www.nd.edu/ccl/software/). The worker count may fluctuate throughout the computational cycle and workers will process tasks until the completion of the computational workload.

Workers can also be submitted to multiple batch systems. Additionally work queue can harness workers from multiple batch systems simultaneously (Bui et al., 2011; Rajan et al., 2011; Moretti et al., 2012). Additionally workers can be run as standalone processes on any machine so it is not necessary to have a batch system available to utilise individual computing nodes. This enables labs of various sizes from very small to large to leverage the computational resources to attack their annotation problems. The general architecture of a work queue application is presented in Figure 4.

The first major issue with combining MAKER and work queue is that MAKER is implemented in Perl and work queue in C. Therefore to simplify the combination of these tools we created a Perl module to interface with the work queue implementation. This was simplified by using SWIG (Simplified Wrapper and Interface Generator, see http://www.swig.org/index.php) and creating an interface file that is input to SWIG and defines the functions and variables present in the C program. Once we had a Perl module that was a working work queue, we needed to modify MAKER to utilise it.

We needed to modify the MAKER source code along with creation of an executable with some logic to be run on the worker nodes. The modification of the MAKER source was primarily the addition of logic to call the work queue module for distribution of the computation. This involved setting up the work queue to accept tasks and then submitting the tiers as tasks to the work queue. We could potentially handle errors in the tier computation on the workers by signalling failure to the work queue that would automatically put the task back into the queue for recomputation. However, we instead return a status to the master node which then can determine which course of action to take in the case of failure, including resubmission of the tier to the queue. The stock MAKER source code is 300 lines of Perl (including whitespace) and our modified version is 555 lines of Perl (including whitespace).

In addition to the benefits produced by distributing the computation across various resources, we also utilise other features of the work queue package such as generating an log file that can easily be parsed to generate visual representations of the path of computation and observe what occurred (e.g. long tail effects). This is often very valuable debugging information for a distributed environment.

MAKER automatically splits the non-genomic inputs into chunks not larger than 100 kb and then combines the output results. However this is not the case for genomic input. Instead it simply splits the genomic data into individual contigs. This means that we have an uneven computational problem. The computation time of chunks is directly related to the size of the input. This makes for a difficult distributed computational problem. We utilise the fast abort functionality of the work queue to enable to abort abnormally long computation on problematic nodes in the distributed environment.

The authors of MAKER provide a few compute times (Cantarel et al., 2008). They provide their analysis on a 2.236 Mb sequence on a 32 GB-RAM machine using a single processor on a machine with eight dual-core 2 GHz processors. They measure the compute time for the full MAKER pipeline at 549 min on one processor and 299 min on two processors. To perform our tests, we retrieved the *C. japonica* genome from Wormbase (Stein et al., 2001) along with protein sequences. Additionally, we retrieved approximately 30,000 EST sequences and assembled them with CAP3 (Huang and Madan, 1999). The genome is composed of 18,817 contigs (with a total length of over 166 MB), the total protein sequences is 36,105 (10 MB) and the assembled EST data is 10,000 contigs (over 5 MB).

These were run through MAKER using control files set-up to use SNAP and est2genome as predictors. All other MAKER options were held at their default settings. The serial version of MAKER was run utilising a single core on a 12-core AMD Opteron machine with 64 GB RAM. The master for the work queue version was run on the same machine with workers being submitted to our campus Condor (Thain et al., 2002) pool. The serial version required over 778 h to complete; while the parallel version run on 50 nodes completed in 17 h for a total compute time of 850 h. This gives a speed-up of 45x, which is less than the ideal linear speed-up of 50x, but is still an good result on a distributed system such as Condor.

Our modifications to the MAKER platform have been provided to the author's of MAKER and are available through their website: http://www.yandell-lab.org/software/maker.html

Acknowledgements

This work was supported in part by National Institutes of Health/National Institute for Allergy and Infectious Diseases grant number HHSN272200900039C and National Science Foundation grant CNS-0643229. Publication of this paper was supported by National Institutes of Health/National Institute for Allergy and Infectious Diseases grant number HHSN272200900039C.

References

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990) 'Basic local alignment search tool', *Journal of Molecular Biology*, Vol. 215, No. 3, pp.403–410.

Besansky, N. (2008) *Genome Analysis of Vectorial Capacity in Major Anopheles Vectors of Malaria Parasites*, White Paper.

Blankenberg, D., Kuster, G.V., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., Nekrutenko, A. and Taylor, J. (2010) 'Galaxy: a web-based genome analysis tool for experimentalists', *Current Protocols in Molecular Biology*, doi: 10.1002/0471142727.mb1910s89.

Bui, P., Rajan, D., Abdul-Wahid, B., Izaguirre, J. and Thain, D. (2011) 'Work Queue+ Python: a framework for scalable scientific ensemble applications', *Workshop on Python for High Performance and Scientific Computing at Supercomputing*, University of Notre Dame.

Burge, C. and Karlin, S. (1997) 'Prediction of complete gene structures in human genomic DNA', *Journal of Molecular Biology*, Vol. 268, No. 1, pp.78–94.

Cantarel, B.L., Korf, I., Robb, S., Parra, G., Ross, E., Moore, B., Holt, C., Sánchez Alvarado, A. and Yandell, M. (2008) 'MAKER: an easy-to-use annotation pipeline designed for emerging model organism genomes', *Genome Research*, Vol. 18, No. 1, pp.188–196.

Dean, J. and Ghemawat, S. (2008) 'MapReduce: simplified data processing on large clusters', *Communications of the ACM*, Vol. 51, No. 1, pp.107–113.

Delcher, A.L., Harmon, D., Kasif, S., White, O. and Salzberg, S.L. (1999) 'Improved microbial gene identification with GLIMMER', *Nucleic Acids Research*, Vol. 27, No. 23, pp.4636–4641.

Durbin, R.M., Altshuler, D.L., Abecasis, G.R., Bentley, D.R., Chakravarti, A., Clark, A.G., Collins, F.S., De La Vega, F.M., Donnelly, P. and Egholm, M. and others (2010) 'A map of human genome variation from population-scale sequencing', *Nature*, Vol. 467, No. 7319, pp.1061–1073.

Emrich, S.J., Barbazuk, W.B., Li, L. and Schnable, P.S. (2007) 'Gene discovery and annotation using LCM-454 transcriptome sequencing', *Genome Research*, Vol. 17, No. 1, pp.69–73.

Foster, I. (1995) *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, Addison-Wesley.

Gentzsch, W. (2001) 'Sun grid engine: towards creating a compute power grid', *CCGRID'01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, pp.35–36.

Giardine, B., Riemer, C., Hardison, R.C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I. and Taylor, J., Miller, W., Kent, W.J. and Nekrutenko, A. (2005) 'Galaxy: a platform for interactive large-scale genome analysis', *Genome Research*, Vol. 15, No. 10, pp.1451–1455.

Goecks, J., Nekrutenko, A., Taylor, J. and Team, T.G. (2010) 'Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences', *Genome Biology*, Vol. 11, No. 8, pp.R86.

Green, P. (1994–1999) *Cross_match*, University of Washington, Seattle, USA. Available online at: http://www.phrap.org/phredphrap/phrap.html

Huang, X. and Madan, A. (1999) 'CAP3: a DNA sequence assembly program', *Genome Research*, Vol. 9, No. 9, pp.868–877.

Korf, I. (2013) *SNAP: Semi-HMM-based Nucleic Acid Parser*, Ian Korf homepage. Available online at: http://homepage.mac.com/iankorf/

Lanc, I., Bui, P., Thain, D. and Emrich, S. (2011) 'Adapting bioinformatics applications for heterogeneous systems: a case study', *Proceedings of the 2nd International Workshop on Emerging Computational Methods for the Life Sciences*, pp.7–14.

Lawson, D., Arensburger, P., Atkinson, P., Besansky, N.J., Bruggner, R.V., Butler, R., Campbell, K.S., Christophides, G.K., Christley, S., Dialynas, E., Hammond, M., Hill, C.A., Konopinski, N., Lobo, N.F., Mac-Callum, R.M., Madey, G., Megy, K., Meyer, J., Redmond, S., Severson, D.W., Stinson, E.O., Topalis, P., Birney, E., Gelbart, W.M., Kafatos, F.C., Louis, C. and Collins, F.H. (2009) 'VectorBase: a data resource for invertebrate vector genomics', *Nucleic Acids Research*, Vol. 37, suppl 1, pp.D583–D587.

Moretti, C., Olson, M., Emrich, S. and Thain, D. (2009) 'Highly scalable genome assembly on campus grids', *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, ACM, New York.

Moretti, C., Thrasher, A., Yu, L., Olson, M., Emrich, S. and Thain, D. (2012) 'A framework for scalable genome assembly on clusters, clouds, and grids', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, No. 12, pp.2189–2197.

Parra, G., Bradnam, K. and Korf, I. (2007) 'CEGMA: a pipeline to accurately annotate core genes in eukaryotic genomes', *Bioinformatics*, Vol. 23, No. 9, pp.1061–1067.

Rajan, D., Canino, A., Izaguirre, J.A. and Thain, D. (2011) 'Converting a high performance application to an elastic cloud application', *IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom'2011)*, pp.383–390.

Salamov, A.A. and Solovyev, V.V. (2000) 'Ab initio gene finding in Drosophila genomic DNA', *Genome Research*, Vol. 10, No. 4, pp.516–522.

Slater, G. and Birney, E. (2005) 'Automated generation of heuristics for biological sequence comparison', *BMC Bioinformatics*, Vol. 6, No. 1, doi:10.1186/1471-2105-6-31.

Stajich, J.E., Block, D., Boulez, K., Brenner, S.E., Chervitz, S.A., Dagdigian, C., Fuellen, G., Gilbert, J.G., Korf, I., Lapp, H., Lehvaslaiho, H., Matsalla, C., Mungall, C.J., Osborne, B.I., Pocock, M.R., Schattner, P., Senger, M., Stein, L.D., Stupka, E., Wilkinson, M.D. and Birney, E. (2002) 'The Bioperl toolkit: Perl modules for the life sciences', *Genome Research*, Vol. 12, No. 10, pp.1611–1618.

Stanke, M. and Waack, S. (2003) 'Gene prediction with a hidden Markov model and a new intron submodel', *Bioinformatics-Oxford*, Vol. 19, No. 2, pp.215–225.

Stein, L., Sternberg, P., Durbin, R., Thierry-Mieg, J. and Spieth, J. (2001) 'WormBase: network access to the genome and biology of caenorhabditis elegans', *Nucleic Acids Research*, Vol. 29, No. 1, pp.82–86.

Thain, D., Tannenbaum, T. and Livny, M. (2002) 'Condor and the grid', *Grid Computing*, pp.299–335.

Thrasher, A., Carmichael, R., Bui, P., Yu, L., Thain, D. and Emrich, S. (2010) 'Taming complex bioinformatics workflows with Weaver, Makeflow, and Starch', *2010 5th Workshop on Workflows in Support of Large-Scale Science (WORKS)*, pp.1–6.