Title goes here

SERVESH MURALIDHARAN, Trinity College Dublin
DAVID GREGG, Trinity College Dublin

Abstract goes here

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

ACM Reference Format:
Muralidharan, S., and Gregg, D., 2012. Title goes here. ACM Trans. Architec. Code Optim. 9, 4, Article 39 (March 2010), 4 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

The constant advancement in server hardware has made systems with large number of hardware threads[] readily available. In addition to this the presence of co-processors[], network processors[], Graphics Processing Units (GPUs) & General Purpose GPUs (GPGPUs) has dramatically improved the capability of commodity hardware. These developments has paved way to a platform with immense computing potential which existing applications already take advantage of. Along with the improvements in the Multi-Core & Many-Core processor architecture the memory and I/O speeds have also seen significant improvement. In particular the network subsystem has seen significant growth, in combination with the improvements to optical communication it has lead to 10GbE interfaces being quite common in enterprise servers. While 40GbE and 100GbE are available for backbone infrastructure but would eventually follow into commodity hardware.

Network applications represent those applications that operate on any of the seven network layers described in the OSI Model. {describe more here}

Several previous work have discussed the process of parallelizing network applications[][][] to enable it to run faster in such multi threaded systems. The development of Software routers[][][] is encouraged further because it costs only a fraction of the commodity hardware routers. Moreover the advancements in the network interfaces has made faster interfaces readily in combination with the multi core servers the performance It has been shown that such routers can be used to process packets at line rate [][][] which has been a challenge on its own. Network applications including those that operate on few or all of the network layers have been designed with sequential processing as the target environment. The shift to multi core architectures in recent times has induced such applications to utilize parallel processing as a alternate means

This work is funded by the IRCSET Enterprise Partnership Scheme along with IBM Research Ireland.
Author's addresses: S. Muralidharan and D. Gregg, Computer Science Department, Trinity College Dublin, Ireland.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

of computing. While this could lead to significant improvement in performance it also suffers from the same problems as any parallel application. These include scalability, load-balancing, data dependency and the presence of non parallel regions in the program apart from those that might arise due to that specific application. The scaling of software routers has been discussed extensively in RouteBricks[] which shows the problems suffered by software routers when they are scaled beyond a single system. One could argue that complexity of network applications such as Deep Packet Inspection (DPI), Packet Forwarding, etc., could be handled by a single many socket Multi-Core SMP system. However, as the complexity of network applications develops it would be inevitable but to move to a multi node environment in order to perform the computations and to handle more network bandwidth.

Large scale computing clusters has been used in high-performance scientific application to solve problems and applications of very large size. It uses a cluster of compute nodes and distributes the workload across them in order to solve it faster. These systems utilize parallel programs consisting of both threads and MPI to break massive problems into smaller workload that can be executed in parallel. In addition, the presence of GPUs and GPGPUs requires a mixture of parallelization constructs such as OpenCL, CUDA, etc., in order to achieve the best performance in the system. In this paper we discuss how some of the techniques used by these applications could be used to parallelize network applications.

Click Modular Software Router[] consists of extensive elements which are used to construct software routers. Click primarily targets packet processing applications especially those that operate on L4 - L7. {elaborate more here}

Discuss about what each sections contains here.....

2. BACKGROUND

Talk about click...

Describe about what would be the best way to parallelize using threads and MPI.. provide reference..

3. DESIGN

This section describes the design principles related to our methodology. The objective is to determine the requirements necessary to parallelize network applications. Our methods focus on ways to improve the click system in some cases even extend its functionality.

3.1. Emphasize on user level execution environment

The existing software routers are predominantly implemented in the kernel space for the ease of access to resources and can bypass the OS overhead incurred when data is accessed through the network subsystem. While this alternative provides better performance it also leads to concerns regarding stability and security of the system. Newer techniques such as Netmap[] allows direct access to network buffers which makes it possible to write high performance packet processing applications in userspace itself. Typically network applications which operate on L4 - L7 are developed in the kernel space in order to intercept the network packets before it flows through the OS. Whereas those applications that utilize L1 - L3 are present in the user space. The objective here is to completely switch to user space to solve any of the stability or security concerns. In addition, scenarios where the applications that operate across most or all the layers could be improved by combining the operations across the different layers in user space itself.

3.2. Eliminate the overhead associated with creation of dynamic objects

Click constructs application graphs based on dynamic creation of objects and passing the data packets across the virtual functions. Though this is a very elegant approach in creating the software router it does incur the penalty of the virtual function tables. The click-tool[] addresses this issue but it does not solve the problem entirely. The construction of the software router at run time is the primary reason behind it. This mainly arises due to the presence of kernel module which installs the router as a kernel module based on the input configuration. However in the user level it would not be necessary for a separate process to initialize the router instead we could generate the entire program statically including the necessary flow associated with the specific application. This not only eliminates the overhead associated but also opens up means to optimize the graph further.

3.3. Optimize based on application graph

Network applications especially that operate on the lower layers are very sensitive to the overhead associated with operating on the data separately on each layer. Instead given an application graph if we were to provide a sequential flow of the different functions at the various layers then it would be possible for the compiler to optimize the regions of program into a single and optimized block of code. This would also improve the cache locality of the data accessed by consecutive functions. This also presents us with the ability to perform other types of stream graph optimizations[] which would not be possible if the specific graph was constructed dynamically at run time.

3.4. Automated data parallelism

Network traffic is a classic example by which data parallelism can be utilized to achieve scalable parallelism. Click's support for parallelism in user level is described as experimental, it is implemented by using a task graph consisting of each work associated with each element which is executed in the threads. The support for parallelism in the kernel mode is better in comparison to that of user mode. Since our target environment is in the user space the proposed design involves a different approach to which we can extract data parallelism. We first split the data across multiple queues and then replicate the application graph across to process the data the queues in parallel. An important condition here is to make sure packets of the same flow should pass through the same queue. Since the application graph can be replicated it also makes it possible to dynamically load balance several applications based on the workload.

3.5. Efficient implementation of task parallelism

In generating the application graph as a sequence of operations it also makes it possible to process different stages of the operation in a pipelined manner. This could be combined with the stream optimizations and could be used to determine which steps of the graph is best suited to form the stages of a pipeline. The condition to balance here is the weight associated with computation in that stage with that of the overhead incurred in computing the data in two separate stages. It would be necessary to determine whether the additional over head involved in executing the task in parallel does provide any significant improvement in performance.

4. IMPLEMENTATION

–Click was designed as a means to easily implement a software router on top of a Linux OS. Click achieves this by implementing it in two different ways one is by installing a kernel module and the other is a program that executes in the userlevel.

–Talk about MPI & threads

–Our goal here is to focus on the network application as a workload and characterize it as such without involving any kind of overhead that might occur due to the network.

5. EXPERIMENTAL EVALUATION

6. RELATED WORK

7. CONCLUSIONS

ACKNOWLEDGMENTS

The authors would like to thank ...