DOCUMENT NO.	REVISION NO.	STATUS
725-xxxx-0001	1.1	**RELEASED**
DEPARTMENT	AUTHOR/CONTACT	EMAIL
Engineering	Andrew Sharp	andy.sharp@onstor.com

Function Specification:

TuxRx: Linux on the TXRX Processor

Approvals:

Name/ Title	**ECO**
Brian Stark/ VP of Cool	

Abstract

Replace EEE based software on Sibyte 1480 processor on Cougar platform with Linux based approach.

ONStor™
Storage Solutions for the Content Generation

Contributors

Name	Email
A. Sharp	andy.sharp@onstor.com

Disclaimer Notice

Copyright Notice

Revision History

Rev	ECO	Written By	Page/Sect	Revision Summary	Date
V0.1		A. Sharp	All	Draft	
V0.2		A. Sharp	All	Incorporate review comments	01/13/09
V1.0		A. Sharp	All	Incorporate review comments; additional clarifications	Jan 19, 2009
V1.1		A. Sharp	All	Incorporate review comments; additional clarifications	Jan 26, 2009

Table of Contents

1 Related Documents

System-X Networking Document:

http://intranet.onstor.net/md/software/systemx/Component_Docs/SystemX_IP_Networking.doc

Illustration 1: TuxRx: The Doctor is IN

2 Relation to Requirements

This project seeks to, in part or in full, address the following Marketing requirements.

- Increased performance
- 10 Gigabit capability
- SMP transport protocols
- Hardware independence
- IPv6
- iSCSI
- DMIP compression, ToS, QoS

3 Relation to Roadmap

The Cougar platform has to have a product life of at the very least two years. The current Cougar software isn't in a position to be relevant for that time frame when new features, increased performance and increased reliability are considered. This project seeks to address that roadmap issue.

In addition, this is an initial foray on implementing System-X. There are other initial forays also in development, and there will be still more others. A full fledged System-X will be the eventual result.

Follow-on efforts are expected to be in the areas of multithreading our file sharing protocols, NFS and CIFS, as well as the other development work necessary to fully deliver iSCSI and IPv6.

4 Problem Statement

Revenue: we have two processor cores on the TXRX 1480 node that currently are sitting idle, leaving capability, and therefore revenue, on the floor with every Cougar we ship.

Converting our product implementation over to System-X in one giant step is never going to be realistically possible. I will call that the System-X roadblock.

There is no practical way to implement a multi-threaded version of our TCP/IP code, yet such a thing is needed for System-X as well as better performance of the current Cougar hardware.

New protocol features: there is no viable way to add additional protocols like IPv6, iSCSI, FTP, or DMIP compression, ToS and QoS, to EEE for a reasonable cost.

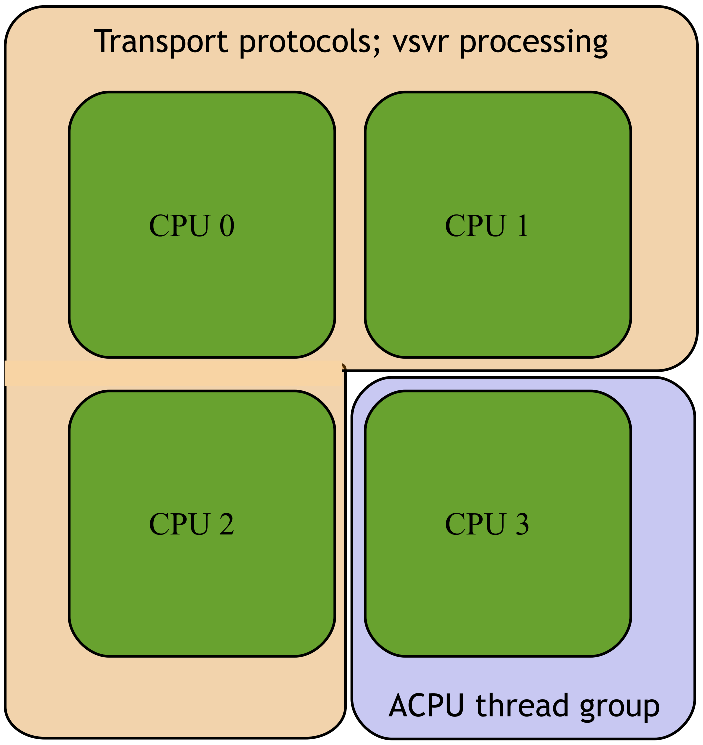
In a similar vein, adding 10 GigE to our EEE based product would cost more than the return on sales of such a feature in the near term, and it would not be well received by customers considering the performance limitations it would have: little better than our current performance with 4 * GigE.

5 Solution

5.1 Software Layout

The solution to the System-X roadblock is to put together a number of smaller steps. This project is one of those steps, and it allows us to take advantage of what comes with it to also solve other long lingering Marketing requirements, like 10 GigE and so forth. In addition, it allows us to actually start making use of the two cores on the TXRX processor that are sitting idle in our current shipping Cougar product implementation, leaving capability on the floor that we could offer our customers and thereby increase our revenue.

The scope of this project is to take an initial step to converting our TXRX software over to something that begins looks like System-X. In this first project, we will put Linux on the TXRX processor, running all four cores as a single SMP incarnation. Our file sharing protocols, NFS and CIFS, often referred to as the ACPU, will be implemented as a thread group (possibly only containing one thread), and will be pinned to a single core. All other threads will be excluded from running on that core. This will allow us to make the ACPU retain its polling architecture without being disturbed by the vagaries of the kernel scheduler, as this thread group will be the only one runnable on its core, so it will never be preempted. Interrupts will be similarly directed to CPU cores as appropriate for the proper performance of the software.



The Linux transport protocols will be used, allowing us to spread out their associated processing work load across the other three cores, thereby, we believe, eliminating some of the bottlenecks in the current implementation and affording us more overall performance.

In follow-on projects, the NFS and CIFS code will be made multithreaded, as well as many of the System-X networking features and design that won't be possible in this first step.

5.2 Memory Layout and Allocation

EEE uses a static memory allocation scheme that is highly appropriate for the I/O task at hand. Linux being a general purpose OS, its memory allocation is much more dynamic. Memory allocation in an OS like Linux can be highly complex because of the interaction between addressing, memory allocation and so forth. Because of that, an attempt to concoct a hybrid scheme for TuxRx is likely to handcuff the network processing, and hence the plan is to convert ACPU to utilize standard Linux memory allocation facilities, except where it relates to data passing with the FP. Standard Linux networking data structures are very similar to the ones used in ACPU, so this is expected to be a relatively straight forward exercise.

In theory this means that both the ACPU and the network protocols will have more memory available for their use than they currently do under EEE, just not at the same time. In I/O centric applications such as ours, this kind of dynamic allocation has historically been associated with memory thrashing problems, hence the design of memory allocation in EEE.

Consequently, there has been some concern mentioned about not being able to quantify ahead of time what kind of impact this implementation change might have on performance, however it is not expected to have an overall negative impact. Actual testing will be required; if it does prove to have a negative impact, we can fairly easily go to a scheme where a chunk of memory is set aside for the ACPU thread to utilize without going through the Linux kernel’s memory allocation facility.

5.3 Debugging and Support

1. Core dumps
   No kernel core dumps are projected at this time, instead stack traces as we do currently on the SSC will be utilized. It is not expected that the kernel will require any real debugging: indeed, that it doesn't is one of the corner stones of this project. The ACPU thread will make use of task core dumps as we utilize in user-space tasks. There will be some user space daemons running on the TuxRx as well which will need to do core dumps.

   FP core/code dumps will be unaffected.
2. Rcon and console commands
   Rcon to the TXRX in its current incarnation will be available to the ACPU thread similar to how it works today. In addition to that, there may be some new Onstor specific utilities in the TuxRx user-space as well which are unique to the TuxRx environment. The **nfsperf**, **req** and **eee** console commands will likely be there in some form, as well as being available for the ACPU in their current form. For example, some memory usage data will be more readily and conveniently available via the the pseudo file /proc/slabinfo. Yes, you may have to log into the TuxRx environment, but we will explore optimizations for that process: proxies; short-cut command; etc.

   Rcon to the FP will be unaffected.
3. Debugging paradigm shift.
   This project represents a significant paradigm change for engineers doing analysis and debugging, and not just because of the existence of a user-space. While some things of the past go away, several more new ones open up. For instance, gdb can be utilized from user-space to analyze certain kernel goings ons, without requiring a special debug build of the kernel. Kernel FTRACE and other such tracing facilities will be available in the later stages of this project, again without a special build or performance impact when not in use. KGDB will be implemented as well.

   There will be a learning curve impact, and the onus will be on the project engineers to help their co-workers in this area.

6 User Interface

There are no anticipated user interface changes of any large significance beyond those already mentioned. Reference section 5.3 concerning rcon and logging into the TuxRx environment.

6.1 Usability Considerations

The only usability requirement is that there be no negative impact on usability. It is expected that use ability will be improved slightly do to the additional accessibility to the kernel from user space via facilities such as /proc and /sys.

7 Dependencies

At this time there are no known dependencies.

7.1 EMRS Dependencies

None at this time. Some modification of the underlying implementation of EMRS may be required, but will be handled by the development team as per As It Should Be.

8 Migration Strategy

There are no known migration roadblocks. At this time it is unclear if the resulting software will be sold as a new product model or merely arrive as just an upgrade. Possibly one or two cores can be turned off for the same model number as we currently ship -- 6700 and 3700 -- and a new model number designated for the 4 core version.

9 Testing Strategy

The testing strategy will be two fold. Since this change will have to be part of a serious release, there should be a lot of testing happening in QA for other reasons.

Additionally, the following testing points will deserve special attention:

- virtual server functionality
- general networking functionality on FP ports
- DMIP
- VLAN functionality
- port bonding
- performance testing: SPEC and streaming
- uptime testing
- limits testing

The development plan for this project includes considerable development of testing code. The plan calls for such testing code to test and exercise the following areas:

- Message passing and handling from the SSC to the TUXRX.
- Virtual server functionality
- Network configuration and functionality
    - Interface creation, enable, disable
    - Bonding
    - VLAN
    - Routing
    - Streaming performance
    - Pkts/sec performance
- TCP/IP relay over management bus

9.1 Key Limits

N/A

9.2 On Disk Formats

No changes to on disk formats will be incurred. Correctness verification will be performed by porting the NFS perftest program from EEE to Linux.

9.3 Cluster Operations

This project will have no effect on cluster operations.

9.4 Cooperative Testing

This project will affect DMIP and DMIP testing will be included as part of the testing strategy.

10 Performance Criteria

Measurements of ops performance during SPEC runs indicated that the limiting factor for Cougar at the time was the TXRX processor. It is expected that adding 2 more CPU cores to job of processing the TCP/IP protocols will alleviate some of that bottleneck and afford an increase in the SPEC ops performance as well as streaming performance with the addition of 10GigE hardware. Exactly how much that increase will be is hard to calculate precisely, but it is hoped to be as much as a 10% increase in SPEC ops, and ideally more than 25% in the streaming performance.

The performance of the ACPU is expected to remain very close to the current profile. This is because, although it will be living under the Linux scheduler, it will be the only thread runnable on its core, so it should never be preempted or rescheduled. The ACPU polling loop will remain as it is currently.