Gokul Nadathur S

825, E.Evelyn Avenue, 638

Sunnyvale, CA 94086

Ph: 408 733 4955

gokul@cs.wisc.edu

Objective

- Seeking a challenging position in software design and development

Education

- M S in Computer Science, University Of Wisconsin Madison, Dec 2000.
- Bachelor of Technology in Computer Science and Engineering,

  Institute of Technology, BHU India, May 1999.

Skills

- Languages: C, C++, TCL
- Platforms: VxWorks, UNIX(Linux, Solaris) , MS Windows
- Protocols: TCP/IP internals, TCP/IP socket programming, Ethernet 802.1d Spanning tree

Work Experience

- May 2002 - Present: Software Engineer at *KuoKoa Networks*, Sunnyvale, CA. Design and development of caching and recovery for a distributed file system
- Jan 2001 March 2002: Software Engineer at *Desana Systems*, Milpitas, CA. Design and development of high performance application for an application level monitoring and control switch
- May 2000 July 2000: Summer Intern, *Desana Systems*, Milpitas, CA. Developed a Virtual Packet Processor and worked on 802.1D spanning tree protocol

- May 1998 July 1998: Summer Intern, National *Aerospace Laboratories(NAL)* India. Implementation of Dual port Memory Controller for Inter processor Communication.

Work Related Projects

- ***Caching and Recovery in Distributed File Systems***
  - Designed and Implemented a Transactional Caching Engine. In addition to being a traditional cache, the caching engine enables clients to group a set of updates and guarantees transactional semantics on this group. The module interacts with the journaling module to provide recoverable semantics on committed transactions.
  - Provided Inputs to design of NVRAM based journaling module. This module provides logging and recovery for transactions initiated by the cache.
  - Designed and Implemented recovery Coordinator for distributed recovery. The function of this module is to initiate recovery as and when different nodes in the system go down and come up without violating the overall data consistency.
  - All the three modules mentioned above interact closely to provide system wide caching and recovery.
  - Proposed and implemented various Cache Optimizations tailored to the Distributed file system to reduce latency of file system operations.
  - Designed and implemented a file system interface (byte level) to a block stream. This facilitated porting of applications that already used a file system interface to use the underlying block stream.
  - Designed and Implemented read write locks with low memory footprint using features in the VxWorks Kernel.
  - Development Environment: x86,MIPS / VxWorks / C

- ***High Performance TCP offload engine***
  - Integrated BSD based TCP/IP stack with Desanas high performance user-space DMA messaging architecture. Re-architected TCP/IP stack to a single threaded asynchronous non-blocking model. Resulting design had zero copies and zero context switches.
  - Used the stack in a TCP/IP proxy application. The application interfaced with control software applications, which performed services like classification, traffic shaping and server load balancing.
  - Improved stack/proxy scalability with a variety of enhancements, including hash based timer wheels and eliminating unused protocol support.
  - Exploited knowledge of performance programming to improve throughput. Transmit descriptor reuse, data structure pre-allocation and other techniques were used to improve CPU cache behavior.
  - Profiled TCP/IP stack, created custom tools for proxy bring-up and validated against ANVL TCP/IP test suit.
  - Development Environment: x86,PowerPC/Linux/C

· ***Virtual Packet Processor***

o Developed packet processing simulation environment. Simulation environment used raw Linux sockets. Simulation environment provided platform for developing layer 2 bridging protocols. Ported spanning tree protocol to environment and validated against IXIA network test equipment. This simulator provided a means for early integration and test of l2/l3 software, before real hardware arrived.

- Development Environment: x86,PowerPC/Linux/C

- ***Service Management Task***

o Re-architected core functionality of distributed process load balancing application. The application managed distributed processes such as TCP/IP proxy, deep content classification and customer management.

- Development Environment: x86,PowerPC/Linux/C++

- ***Hardware Bring up and Debugging***

  - Designed and implemented an automated test framework for the systems distributed messaging infrastructure. Framework could configure test cases in numerous topologies. This framework accelerated the stabilization effort of the system, by identifying bugs in software and the underlying hardware switch fabric.

  - Development Environment: PowerPC/Linux/C

Academic Projects

- ***Runtime code modification for avoidance of license server authentication:*** The project involved implementing tools that changed the address space of a running process so that the code path that is executed is changed. This was used to prevent the process from attempting to obtain licensing information, yet reflect a state that license has been obtained. The /proc interface was used in building the tools. Development Environment: UltraSparc/Solaris/C

- ***Scalable and fault tolerant middleware for multicast reduction operations in distributed cluster environments:*** The middleware was developed as a library to provide a communication interface for building applications that need to scale with the number of nodes in the cluster. Dynamic code execution techniques (dynamically linkable shared objects) were used to inject new application level

functionality at the nodes. Development Environment: x86/Linux/C

- ***Quilt query Interface for XML data querying:*** The project involved constructing a scanner and parser for the quilt language. After parsing the query, it was translated into an optimized logical plan of the underlying query execution engine. Development Environment: x86/Linux/C++

- ***Reliable Multicast Network/Transport layer Protocol:*** The network layer consisted of a forwarding protocol, link state routing protocol and group management protocol. The transport layer offered reliability and flow control. The stack was then used to broadcast movies to different recipients who formed a multicast group. Development Environment: x86/Solaris/C

- ***Simulation Study of Various Instruction Pre-fetching Techniques:*** Simulated various instruction prefetching techniques such as fetch directed prefetching, stream buffers and studied the performance characteristics of these methods.