Tapping into the Fountain of CPUS-On Operating System Support for Programmable Devices

Yaron Weinsberg¹, Danny Dolev¹, Tal Anker¹, Muli Ben-Yehuda² and Pete Wyckoff³

muli@il.ibm.com

¹ The Hebrew University Of Jerusalem (HUJI)
² IBM Haifa Research Lab
³ Ohio Supercomputer Center
The Elevator Pitch

• Today’s peripheral devices are very powerful
  – Contain general purpose CPUs, memory, specialized hardware
  – Programmable - more flexible than ASIC solutions

• Can we use them in order to execute parts of our OS and user-level applications?
  – Yes, but…
  – There is no generic framework that enables this…
The Elevator Pitch

• Hydra is a generic “offloading framework”
  – Provides a programming model and runtime support that enables one to develop *Offload-Aware (OA) Applications*
  – “Aware” of any available (programmable) computing resource

• Enables a developer to define the offloading aspects of the application during design time
“TivoPC”

You can now compile your kernel while watching the Superbowl...
TivoPC Information Flow

NIC -> GPU -> Disk Controller
Why should we deal with offloading when a typical host is full of CPUs?
Reasons for Offloading

- **Memory Bottlenecks**
  - reduce memory pressure and cache-misses (due to filtering done at the device)

- **Timeliness guarantees**
  - GPOS ↔ Embedded OS (RTOS)
  - avoiding “OS noise” (interrupts, context switches, timers etc.)

- **Reduced power consumption**
  - Pentium 4 2.8Ghz: 68Watt
  - Intel XScale 600Mhz: 0.5Watt
Reasons for Offloading

- **Security**
  - harder to attack
- **Increased Throughput**

*The graphs appear in the paper: “TCP performance revisited” by Foong et. al. (ISPASS’03) and are used with the authors’ permission.*
Outline

• Motivation
• HYDRA Programming Model
• HYDRA Architecture
• Evaluation
• Future Work
The Current Gap

- **Not** many applications **DO** take advantage of the available processing power...
- Using programmable devices has traditionally been very difficult:
  - Requires experienced embedded engineers
  - Requires customization of each particular design for each peripheral device

→ **HYDRA** to the rescue...
Outline

• Motivation
• HYDRA Programming Model
• HYDRA Architecture
• Evaluation
• Future Work
HYDRA Programming Model

- Hydra defines **“Offcodes”** (Offloaded-Code)
  - The minimal unit for offloading
  - Exports a well defined interface (like COM objects)
  - Given as open source or as compiled binaries
  - Described by Offcode Description File (ODF)
    - Exposes the offcode’s functionality (interfaces)
    - Defined the offcode’s dependencies
Offcode Libraries

- Offcode Library
  - Networking
  - Math
  - Graphics
  - Security

Import

OA-App

import

Networking

- BSD Socket
  - socket.odf
- CRC32
  - crc32.odf

User Lib

- mpeg
  - Decoder.odf
Channels

- Offcodes are interconnected via *Channels*
  - Determines various communication properties between offcodes

1. An Out-Of-Band Channel, *OOB-channel*, is attached to every OA-application and Offcode
   - Not performance critical (uses memory copies)
   - Used for initialization, control and events dissemination

![Diagram showing interconnected offcodes via OOB-channel and Specialized channel](image-url)
Channels

(II) A specialized channel is created for performance critical communication

– Hydra provides several channel types:
  – Unicast / Multicast
  – Reliable / Unreliable
  – Synchronized / Asynchronous
  – Buffered / Zero-Copy R/W/Both
Design Methodology

• OA-applications are designed by two orthogonal aspects:

  1. **Basic logic design:**
     Design the application logic and define the components to be offloaded

  2. **Offloading Layout design:**
     Define the channels of communication between offcodes and their location constraints
## 1. Logical Design

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI</td>
<td>Provides the viewing area and user controls (play, pause, rewind and resume)</td>
</tr>
<tr>
<td>Streamer</td>
<td>Process the media stream (either from network or storage)</td>
</tr>
<tr>
<td>Decoder</td>
<td>Decodes the MPEG stream</td>
</tr>
<tr>
<td>Display</td>
<td>Displays the movie on screen</td>
</tr>
<tr>
<td>File</td>
<td>Reads/Writes data from storage</td>
</tr>
</tbody>
</table>
2. Offloading Layout Design

- GUI
- Streamer
- File
- Display
- Decoder

- NIC
- CPU
- Disk Controller
Finally: Application Deployment

Logical Devices → mapping → Physical Devices

Layout Graph → mapping → Offcode Generation

Offcode Generation → Offloading → Execution
Outline

• Motivation
• HYDRA Programming Model
• HYDRA Architecture
• Evaluation
• Future Work
HYDRA Architecture

• The runtime system implements the programming model

• Both the host OS and target devices must implement the runtime functionality
HYDRA Architecture

User

Kernel

Runtime API

Channel Providers

Device

Device OS

Sys Call Mgmt  Offloading API  Channel Mgmt  Memory Mgmt  Layout Mgmt  Resource Mgmt

Channel Executive

Local  Remote

PCI  TCP/IP  iSCSI  RDMA

Offcode

α

β

Offloading Runtime Extensions
HYDRA Architecture

- hydra.ko module
  Hydra Kernel Level Runtime Support
- libhydra.so
  for OA-application developers
- Device’s Hydra Runtime Implementation

Kernel
- Sys Call Mgmt
- Offloading API
- Channel Mgmt
- Memory Mgmt
- Layout Mgmt
- Resource Mgmt

Channel Providers
- Local
  - PCI
  - TCP/IP
  - iSCSI
  - RDMA
- Remote

Device
- Offcode $\alpha$
- Offcode $\beta$

OA-App
- User layer API
HYDRA Architecture

Kernel
- Sys Call Mgmt
- Offloading API
- Channel Mgmt
- Memory Mgmt
- Layout Mgmt
- Resource Mgmt

Channel Providers
- Local
- Remote
- PCI
- TCP/IP
- iSCSI
- RDMA

Runtime API

Device
- Offcode $\alpha$
- Offcode $\beta$

Device OS
- Offloading Runtime Extensions

user layer API
HYDRA Architecture

User
- OA-App
  - user layer API

Kernel
- Sys Call Mgmt
- Offloading API
- Channel Mgmt
- Memory Mgmt
- Layout Mgmt
- Resource Mgmt

Channel Providers
- Channel Executive
  - Local
  - Remote
    - PCI
    - TCP/IP
    - iSCSI
    - RDMA

Device
- Offcode $\alpha$
- Offcode $\beta$

Runtime API

Parsing and processing of the layout graph
HYDRA Architecture

User

Kernel

Kernel Providers

Channel Executive

Device

Device OS

Memory pinning services
Channel, Offcode Factory
HYDRA Architecture

User

- OA-App
- user layer API

Kernel

- Sys Call Mgmt
- Offloading API
- Channel Mgmt
- Memory Mgmt
- Layout Mgmt
- Resource Mgmt

Runtime API

Channel Executors

- Local
- Remote

Channel Providers

- PCI
- TCP/IP
- iSCSI

Device

- Offcode $\alpha$
- Offcode $\beta$

Device OS

- Offloading Runtime Extensions
HYDRA Architecture

Handles Channel Creation via device specific providers

Future support for Hydra’s distribution framework

Channel Provider on the local host
Outline

• Motivation
• HYDRA Programming Model
• HYDRA Architecture
• Evaluation
• Future Work
Evaluation – TiVo-PC

- The server streams 1KB packets, every 5 msec (200KB MPEG movie)
Evaluation - TiVo-PC

Packets Jitter (at the video client)
Evaluation - TiVo-PC
L2 Cache Miss Ratio (Server)

![Bar chart showing L2 Cache Miss Ratio for different server states.](image)

- **Idle**: Slowdown of 1.00
- **Simple Server**: Slowdown of 1.07
- **Sendfile Server**: Slowdown of 1.015
- **Offloaded Server**: Slowdown of 1.00
Outline

• Motivation
• HYDRA Programming Model
• HYDRA Architecture
• Evaluation
• Future Work
Future Directions

- **OS Offloading**
  - File system (NFS, indexing, caching, buffer cache...)
  - Device drivers offload

- **Multi-core support**
  - CMPs, SMPs

- **Security**
  - RNGs, En/Decryption, tripwire, IDS/IPS, firewall

- **I/O for virtualized systems**
  - IOMMUs, pinning
Thanks!