The PUNCH Portal to Internet Computing: Run Any Software Anywhere via WWW Browsers

Nirav H. Kapadia       José A. B. Fortes       Mark S. Lundstrom

School of Electrical and Computer Engineering
Purdue University

I. Introduction to PUNCH
II. System Design Overview
III. External System Interface
IV. Internal Architecture
V. Supported Resources
VI. Resource Management
VII. Conclusions
I. Introduction to PUNCH
II. System Design Overview
III. External System Interface
IV. Internal Architecture
V. Supported Resources
VI. Resource Management
VII. Conclusions
PUNCH Overview:

- PUNCH is a distributed network-computing infrastructure that allows geographically dispersed users to run unmodified tools via standard Web browsers

(www.ece.purdue.edu/punch)
Core Research Team:

Nirav Kapadia, José Fortes, Mark Lundstrom, Renato Figueiredo, and Sumalatha Adabala

Collaborators:

Rudolf Eigenmann, Dolors Royo, and José Miguel-Alonso

Partner Institutions:

800+ users
3,000,000+ hits
users in 10 countries
usage doubles each year

50+ engineering tools
“as is” installation in 30 mins
operational for five years
version 5 being developed
The Purdue University Network-Computing Hubs (PUNCH)

- The Computer Architecture Hub
- The Parallel Programming Hub
- The VLSI Design Hub
- The Semiconductor Simulation Hub
- The DESCARTES Toolkit

Physical Location ‘1’

Tool (Application)
Compute-Server
Parallel Machine
Workstation Cluster

Physical Location ‘n’

Tool (Application)
Compute-Server
Parallel Machine
Workstation Cluster

Interconnected by Internet / Intranet
Portal Screen Shot

(http://www.ece.purdue.edu/punch/)
I. Introduction to PUNCH
II. System Design Overview
III. External System Interface
IV. Internal Architecture
V. Supported Resources
VI. Resource Management
VII. Conclusions
Network-Computing Systems

- External System Interface (external view of the system)
- Internal Architecture (internal system design and structure)
- Resources Supported (class of programs/machines supported)
- Resource Management (application management; resource allocation)
External System Interface:

- **Objective:** universal access
- **Approach:** leverage the World Wide Web
- **Challenge:** work within the existing WWW framework
- **Status:** students and researchers in ten countries routinely use PUNCH via standard Web browsers; 50+ discipline-specific and productivity tools are available
Internal Architecture:

- **Objective:** scalability and interoperability
- **Approach:** hierarchically distributed architecture
- **Challenge:** manage tools and information *in situ*
- **Status:** system deployed on Unix; support for customized user views, global user accounts, and virtual filesystems; root access not required
Supported Resources:

- **Objective:** work with unmodified tools
- **Approach:** virtual interfaces; configurable environment
- **Challenge:** state management; flow control
- **Status:** batch, interactive, and graphical tools supported; new prototype supports PVM programs and Condor, DQS, and PBS jobs
Resource Management:

- **Objective**: address usage policy *and* performance issues
- **Approach**: metaprograms; application management
- **Challenge**: predictive performance modeling; interoperability
- **Status**: scheduling customizable on a per-tool and per-user basis; prototype machine learning system facilitates automated cost/performance tradeoff decisions
I. Introduction to PUNCH
II. System Design Overview
III. External System Interface
IV. Internal Architecture
V. Supported Resources
VI. Resource Management
VII. Conclusions
External System Interface:

- **Problem**: must work within existing WWW framework
  **Solution**: virtual URLs; web-accessible OS

- **Problem**: HTML is static
  **Solution**: HTML templates; embedded variables/objects

- **Problem**: no flow-control
  **Solution**: metaprogams; programmable state machine
WWW-Based Computing (Issues):

<table>
<thead>
<tr>
<th>WWW Framework</th>
<th>WWW-Based Internet/Intranet Computing</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique mapping between URL and some entity within the filesystem (physical URL).</td>
<td>URL is a functional request that is mapped in a context-sensitive manner (virtual URL).</td>
<td>WWW URL model does not support truly dynamic URL mappings.</td>
</tr>
<tr>
<td>Mostly public space, with some support for access control.</td>
<td>Mostly private space, with some support for sharing user data.</td>
<td>Authentication model for WWW does not scale.</td>
</tr>
<tr>
<td>Designed for document serving, with some support for dynamic information.</td>
<td>Primarily provides OS functions (e.g., process management). Document serving is secondary.</td>
<td>Standard WWW resource management model not appropriate.</td>
</tr>
<tr>
<td>Designed for a (mostly) read-only environment.</td>
<td>A read-write environment is critical for correct operation of OS services.</td>
<td>Need synchronization and resource locking. Support for a multi-user web OS.</td>
</tr>
<tr>
<td>Processing supported as an add-on via CGI scripts.</td>
<td>Processing is an integral part of the service.</td>
<td>Integration of distributed computing and WWW.</td>
</tr>
</tbody>
</table>

UPC Presentation, June 2000

Nirav Kapadia, Purdue University
Virtual Addresses:

Translation process involves a simple table lookup.
Virtual URLs:

Translation process involves database queries, and generates side-effects (e.g., state and context initialization).
HTML Templates (Variables/Objects):

Begin PageTemplate
  Begin Declarations
    menu myFiles = 1:<UserFiles>;
    bind myFiles = <WorkingFolder>;
  End Declarations

  Begin HTML
    <FORM>
      Select a file (or folder to open): <myFiles> <P>
      <CENTER><INPUT TYPE="submit" VALUE="Proceed"></CENTER>
    </FORM>
  End HTML
End PageTemplate
Metaprograms (Flow Control):

```plaintext
retrievestate 'directory';
display 'Page1';
while(isdir(<myFiles>))
{
    chdir(<myFiles>);
    display 'Page1';
}
savestate 'directory';
```

- 
- 
-
Performance Summary:

<table>
<thead>
<tr>
<th>Tool Interface</th>
<th>Type of Transaction</th>
<th>Response Time (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No Preforking</td>
</tr>
<tr>
<td>Static Info</td>
<td>Publicly-Accessible URL</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>Access Control Enforced</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>Process Status</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>Dynamic Information</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>Entry Page</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Input Management</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>Run Management</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>Output Management</td>
<td>61</td>
</tr>
</tbody>
</table>

UPC Presentation, June 2000

Nirav Kapadia, Purdue University
I. Introduction to PUNCH
II. System Design Overview
III. External System Interface
IV. Internal Architecture
V. Supported Resources
VI. Resource Management
VII. Conclusions
Internal Architecture:

- **Problem**: dynamic/incremental/distributed information
  **Solution**: hierarchically distributed architecture

- **Problem**: multiple administrative domains
  **Solution**: resources usable/visible/invisible; partitioned root

- **Problem**: scalability of information retrieval
  **Solution**: hashing; access codes
Core Architecture:

User

Client Unit

Management Unit

Execution Unit

Resources

User-Specific Information
(e.g., tool-input, preferences, etc.)

Tool-Specific Information
(e.g., algorithmic complexity,
platforms supported, etc.)

Site-Specific Information
(e.g., default configuration,
path to executable, etc.)
1. User-access via standard WWW browsers.
2. Network Desktop processes and responds to all "non-application-invocation" requests.
3. Application-invocation requests are forwarded to an appropriate Management Unit.
4. Management Unit authenticates request, determines resource requirements, and selects Execution Unit.
5. Execution Unit processes run request and notifies Management Unit on completion.

---

**Diagram Description:**

- **User Platform:** Represents the user interface, with HTTP as the communication protocol.
- **Network Desktop Interface:** Facilitates interaction between the front-end and back-end systems.
- **Management Units:** Handles user authentication and management of resources.
- **Execution Units:** Processes requests and communicates with managed resources.
- **Application Repositories:** Stores application data and resources.
- **Managed Resources:** Includes tools, application specifications, logical resource interface, scheduling, job control, tool-specific input analysis, resource resolution, and resource allocation.

---

**Key Components:**

- **Hub User Accounts:** Signifies user authentication and access control.
- **Network Management Subsystem:** Integrates various tools and resources for efficient management.
- **Tool-Kept Knowledge Base:** Stores and manages tool-specific data and information.
- **View Customization:** Customizes user interface for enhanced usability.
- **Authentication:** Ensures secure user access and interaction.
- **Tool-Specific Input Analysis:** Analyzes and processes tool-specific input data.
- **Resource Resolution:** Allocates and resolves resources efficiently.
- **Resource Allocation:** Manages and distributes resources according to user needs.
Scalable Information Retrieval:

(n Identifiers)

Identifier

Password

Hash Function

Hash-Table Size (k)

Identifier

Hash Index (Filename Extension)

File (k files; ~ n/k entries per file)

Resource-Lookup Manager

Resource

Access Code (Offset in File)

Access Code (Offset in File)

(0(1) access with code)
Effects of Using a Hash Function (994 Users)

- Number of Buckets vs. Average Authentication Time, in milliseconds

Effects of Using Access Codes (994 Users)

- Number of Buckets vs. Average Authentication Time, in milliseconds

UPC Presentation, June 2000
Nirav Kapadia, Purdue University
## Access Code Proliferation

<table>
<thead>
<tr>
<th>Type of Use</th>
<th>Access Code</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Present</td>
</tr>
<tr>
<td>Member Authentication</td>
<td>1,086,732</td>
</tr>
<tr>
<td>Internal References</td>
<td>946,652</td>
</tr>
</tbody>
</table>

UPC Presentation, June 2000

Nirav Kapadia, Purdue University
I. Introduction to PUNCH
II. System Design Overview
III. External System Interface
IV. Internal Architecture
V. Supported Resources
VI. Resource Management
VII. Conclusions
Supported Resources:

- **Problem:** many tools are not network-aware
  **Solution:** configurable “operating system” and environment

- **Problem:** must accommodate independently-developed tools
  **Solution:** resource description language for tools

- **Problem:** graphical user interfaces
  **Solution:** leverage display-management technologies
Compiler

General Information
- Access Control
- Logical Indexing
- Documentation
- MIME Types

Application Management
- Input Translation
- Implementation Selection
- Input Pre-Processing
- Execution Syntax
- Output Post-Processing

Tool Specification

Interface Generation
- Interface Template
- Flow Control
- Input Grammar
- Analysis Rules

UPC Presentation, June 2000
Nirav Kapadia, Purdue University
I. Introduction to PUNCH  
II. System Design Overview  
III. External System Interface  
IV. Internal Architecture  
V. Supported Resources  
VI. Resource Management  
VII. Conclusions
Resource Management:

- **Problem**: usage policies and performance issues intertwined
  **Solution**: metaprograms; application management

- **Problem**: dynamic, heterogeneous environment
  **Solution**: run-time cost/performance tradeoff decisions

- **Problem**: need *a priori* estimates of resource usage
  **Solution**: predictive application-performance modeling
Resource Management Model:

Usage Policies  Access Control  Resource Availability  Resource Status  Scheduling Objective

Administrative Constraints  Performance Constraints

Tool-specific Information  Run-specific Information

Performance Modeling

Demand-driven Resource Management
Resource Management Architecture:

Users

1. PUNCH Network Desktop

2. 

SCION

Application Management

Query Manager

Pool Management Hierarchy

Pool Manager

Pool Manager

Pool Manager

Resource Management Pipeline for Computational Grids

Resource Pool

Users

3. PUNCH Network Desktop

4. 

Managed Resources

5. 

Resource Knowledge-base

UPC Presentation, June 2000

Nirav Kapadia, Purdue University
Parse User-Input

Extract Relevant Parameters

Qualify Extracted Information

Select Appropriate Algorithm(s)

Determine Hardware Requirements

Reserve Resource(s)

More Options?

Y

More Options

Low

Verify Delivered QoS

N

Ok

Re-Negotiate QoS and Repeat/Abort

Dispatch Request

e.g.: Simulate carrier transport for the given device specs; QoS specified in terms of one or more of wall-clock time, Mflop/s, I/O, etc.

e.g.: #carriers

#nodes in grid

device size

convergence norm

e.g.: cpuUnits = f(parameters)

memReqd = g(parameters)

Rank Algorithms:

f(parameters, available algorithms)

e.g.:

Monte Carlo, Hydro-Dynamic, Drift Diffusion

e.g.:

SPARC or HP Architecture with >=256MB RAM and >=300 SPECfp

Get Resource:

f(Architecture, Memory, I/O, Performance, QoS)

e.g.:

SPARC ULTRA (karr.ecn)
Performance Modeling (Synthetic Dataset):
Performance Modeling (Real Dataset):

- Tool: T-Suprem3, a commercial silicon process simulator
- Data: ≈8,100 runs from normal use of PUNCH over 10 months
- Features for T-Suprem3:
  - number of grid points,
  - total diffusion time,
  - cumulative epitaxial growth,
  - minimum implant energy,
  - number of deposit steps,
  - number of etch steps,
  - number of implant steps
Summary of Results:

Results for T-Suprem3 Data-Set

- Run Number
- Prediction Error (s)
- NN, Wtd. Avg., LLWR

- Run Number
- Lookup Time (s)
- noedit, c=0, noedit, c=5, iedit, c=0, iedit, c=5

- Run Number
- #Occurrences
- Exact Match, Linear LWR

UPC Presentation, June 2000
Nirav Kapadia, Purdue University
I. Introduction to PUNCH
II. System Design Overview
III. External System Interface
IV. Internal Architecture
V. Supported Resources
VI. Resource Management
VII. Conclusions
Related Research:

- **Application-specific designs**
  Exploratorium, JSPICE, SIP, Virtual Lab, etc.

- **Systems for numerical software**
  MMM, NEOS, NetSolve, Ninf, RCS, etc.

- **Systems for general-purpose software**
  Bond, Nimrod, PUNCH, Rivendell, WinFrame, etc.

- **Systems for metacomputing applications**
  ATLAS, Globe, GUSTO, IceT, Legion, etc.
Industry Directions:

- New ASPs -- e.g., AppCity and Corio
  ( focus is on business apps, as opposed to computing apps )

- Sun, Microsoft, and others
  ( focus is on new or rewritten applications )

- Software vendors -- e.g., CAD companies
  ( applications are modified to work with browsers )

- Middleware companies -- e.g., Citrix and SCO
  ( closest to PUNCH’s approach; connectivity solution )

- PUNCH infrastructure
  ( deployed and tested over 5 years; version 5; global user base )
Summary: External System Interface

- Access via standard WWW browsers
- Semantics for computing services via virtual URLs
- HTML templates (embedded objects); compiler; generator
- Token-based caching (access codes); user-view customization

Summary: Internal Architecture

- Scalable information management
- Security and access-control across domains
- Account, process, and data management
- Error management and recovery
Summary: Supported Resources

- Tool specification and metaprogramming languages; compilers
- Metaprogramming environment (interpreter, virtual machine)
- State/session management; persistence engine
- Programmable parser (to emulate interactivity)

Summary: Resource Management

- Run-specific resource-usage prediction
- Knowledge management; locality of runs
- Run-time cost/performance tradeoff decisions
- Interoperation with external resource management systems
<table>
<thead>
<tr>
<th>Implications of Network-Based Wide-Area Computing</th>
<th>The Purdue University Network-Computing Hubs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pervasive access to computing for authorized users.</td>
<td>Users access and run tools via standard WWW browsers.</td>
</tr>
<tr>
<td>Ability to work across administrative domains.</td>
<td>Hierarchically distributed architecture; partitionable.</td>
</tr>
<tr>
<td>Dynamic addition and removal of resources.</td>
<td>Resources can be added or removed via configuration files.</td>
</tr>
<tr>
<td>Ability to work with diverse software.</td>
<td>Unmodified tools installed in as little as thirty minutes.</td>
</tr>
<tr>
<td>Ability to determine run-specific “needs”.</td>
<td>Learning algorithms allow resource-usage prediction.</td>
</tr>
</tbody>
</table>