Some Future Challenges of Binary Translation

Kemal Ebcioglu

IBM T.J. Watson Research Center

Disclaimers

- The IBM material in this tutorial consists of non-confidential *research* projects and ideas
- Non-IBM material has been obtained from public sources only

The nature of binary translation

- Binary translation is just-in-time compilation from binary code of one architecture to another
  - HP Dynamo, IBM DAISY, Compaq FX!32, Transmeta Crusoe, Fundamental Software FLEX-ES,…
- Some main attractions
  - Implement architecture as a layer of software– simplifies the hardware
  - Additional optimizations on binary code, not caught by static source level compiler
- Is BT a “Disruptive Technology”?
- BT is a good microarchitecture research topic
  - Could change the way microprocessors are designed

Binary translation vs. traditional compilation

- Radical change in approach needed
- Traditional compiler techniques (e.g. register coloring) can take too long
  - BT must gracefully tolerate low re-use rates
  - Focus on techniques with most “performance return on time investment”
- Exploit optimizations across procedure boundaries
- Exploit run time information for optimization and re-optimization
- Deal with precise exceptions
- Deal with lack of source level information

Difficulties of binary translation

- Challenges that are already mostly addressed
  - Self modifying code, re-ordering memory ops with strong MP consistency, memory mapped I/O, precise exceptions, emulating different “important” virtual memory architectures on same TLB hardware, achieving ILP, …
- Not completely solved
  - Real time deadlines, very low re-use rates, memory wasted in translation cache, …
- There is an ample amount of ways to fail …
  - Many tradeoffs exist, design space is large, experimentation is laborious
  - Best engineers can get stuck in unfruitful regions

A possible future use of BT: Hardware commonality

- Hardware commonality across different architectures
- Architecture as a layer of software
- Different software layers implement different “important” architectures on the same processor core
- Disappearance of software migration difficulties
  - No more captive customers
- If the core and software are not proprietary, and are adopted as a standard:
  - The core architecture can become even more commoditized than x86’s today.
  - Rate of performance increase or power reduction could be higher in the common core– due to more efficient focus.
    - Potential to overtake mainstream
  - Value would still remain in enhancements to the hw/sw

A possible future use of BT: Virtual IT Shop

- The emerging trend is to rent your virtual IT shop over the Internet (e.g. Loudcloud, Corio)
  - Computing resources “on tap”
- Big, secure server farms house the machines
- Need to grow or shrink the number of virtual machines of customer on demand
- But hard to virtualize the heterogeneous architectures in traditional IT shops
  - E.g. if customers would like SPARC, x86, and S/390, in their IT shops, how many would you stock from each in the server farm?

Virtual IT Shop

- Binary translation can solve problem with virtual IT shops with heterogeneous architectures
  - Build racks of identical processors/boards supporting multiple architectures through BT
  - Even have the same core emulate a second virtual architecture when the first is in idle wait
  - Significantly better resource allocation
  - More reliable: when one virtual processor crashes due to the application software, the physical processor hosting it does not.

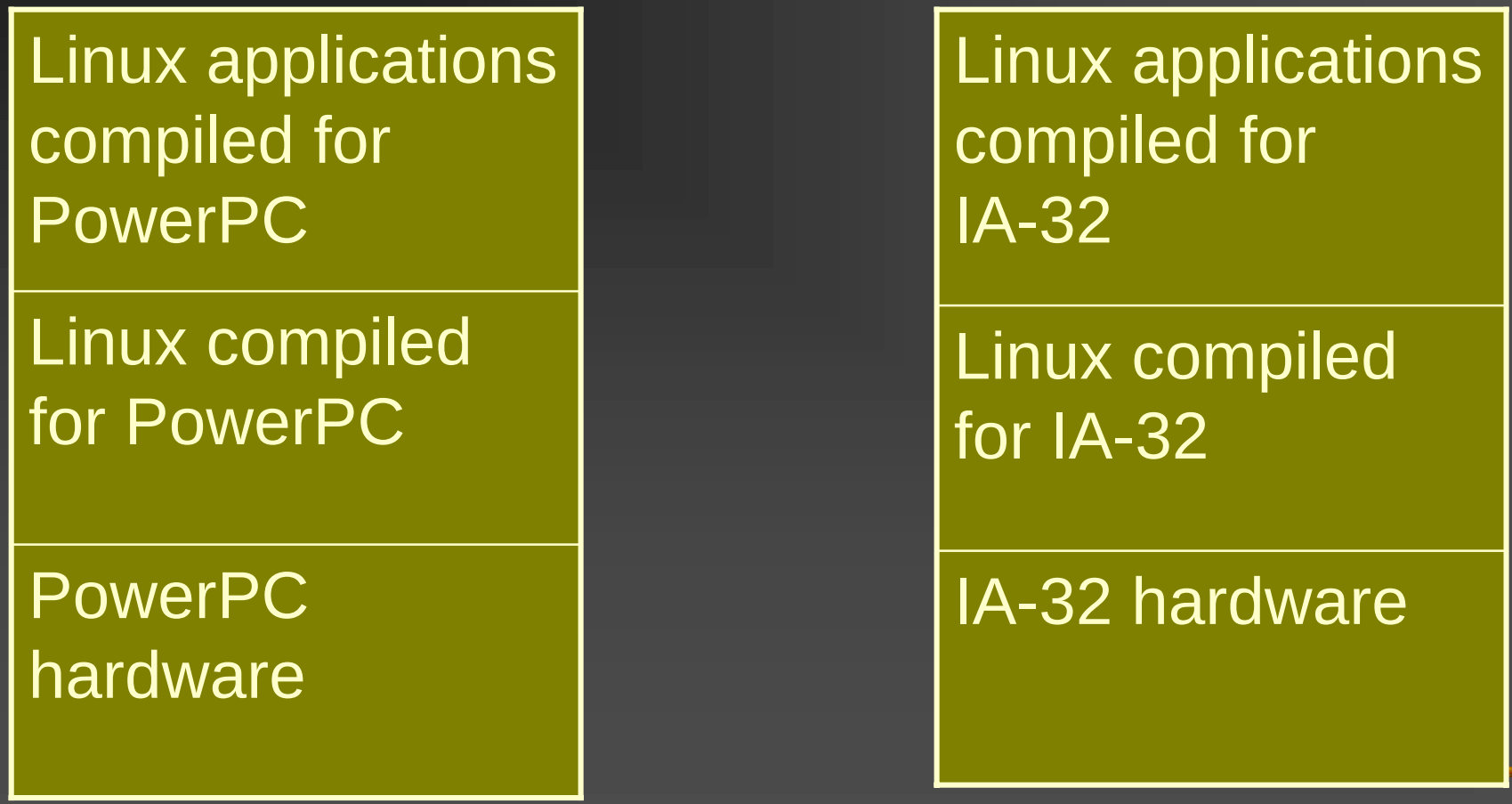
Another future use of BT: Convergence VM

- Internet is changing software landscape toward convergence of architectures
  - Platform independent paradigms: XML, SOAP, Java
  - But true platform independence did not succeed with JAVA
    - JAVA OS was to rival Wintel, but did not gain acceptance
    - You cannot run existing C/C++ apps and OSs on JAVA
  - IBM S/390, PowerPC, AS/400, x86 servers renamed as part of the same series, around e-business
  - Time seems opportune for a new attempt at platform independent software
    - Premise: JIT compilation performance will be equal to or better than static compilation performance

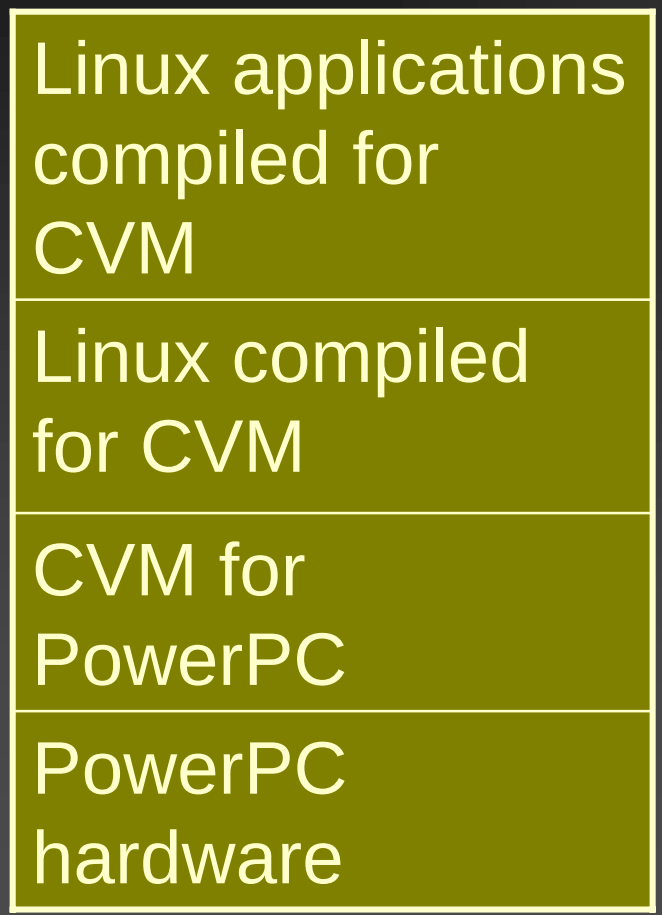
Convergence VM

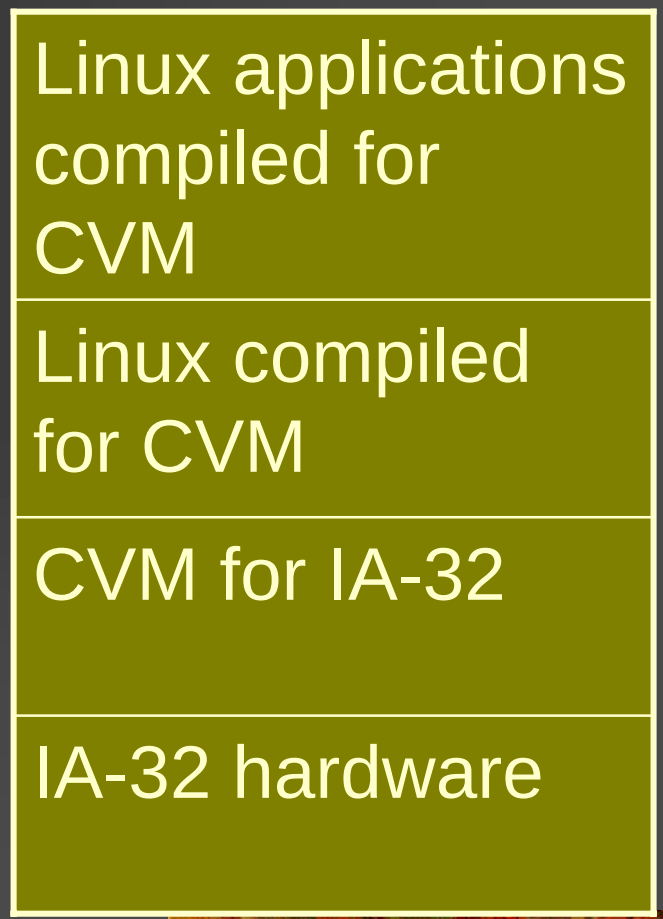
- Define a simple but complete common virtual machine (RISC-like) with standard virtual I/O devices
  - Object code can be in XML format—OSs can be booted from the Web
- Implement VM on multiple architectures with state-of-the-art JIT compilation and runtime device emulation
- Port a standard OS and e-business software to the VM
- Can allow one to run the same OS and application object code on different hardware

Current systems



Possible future systems with CVM





Conclusions

- Binary translation, if successful, has potential to make a game-changing impact
- Microarchitects can focus on using the best architectural features in a common core
- Dynamic translation techniques may ultimately take on more importance than static compilation
- But many challenges exist
  - That is why we think it is interesting research!