Worldwide Developers Conference

Multiprocessing Strategies for Mac OS and Rhapsody

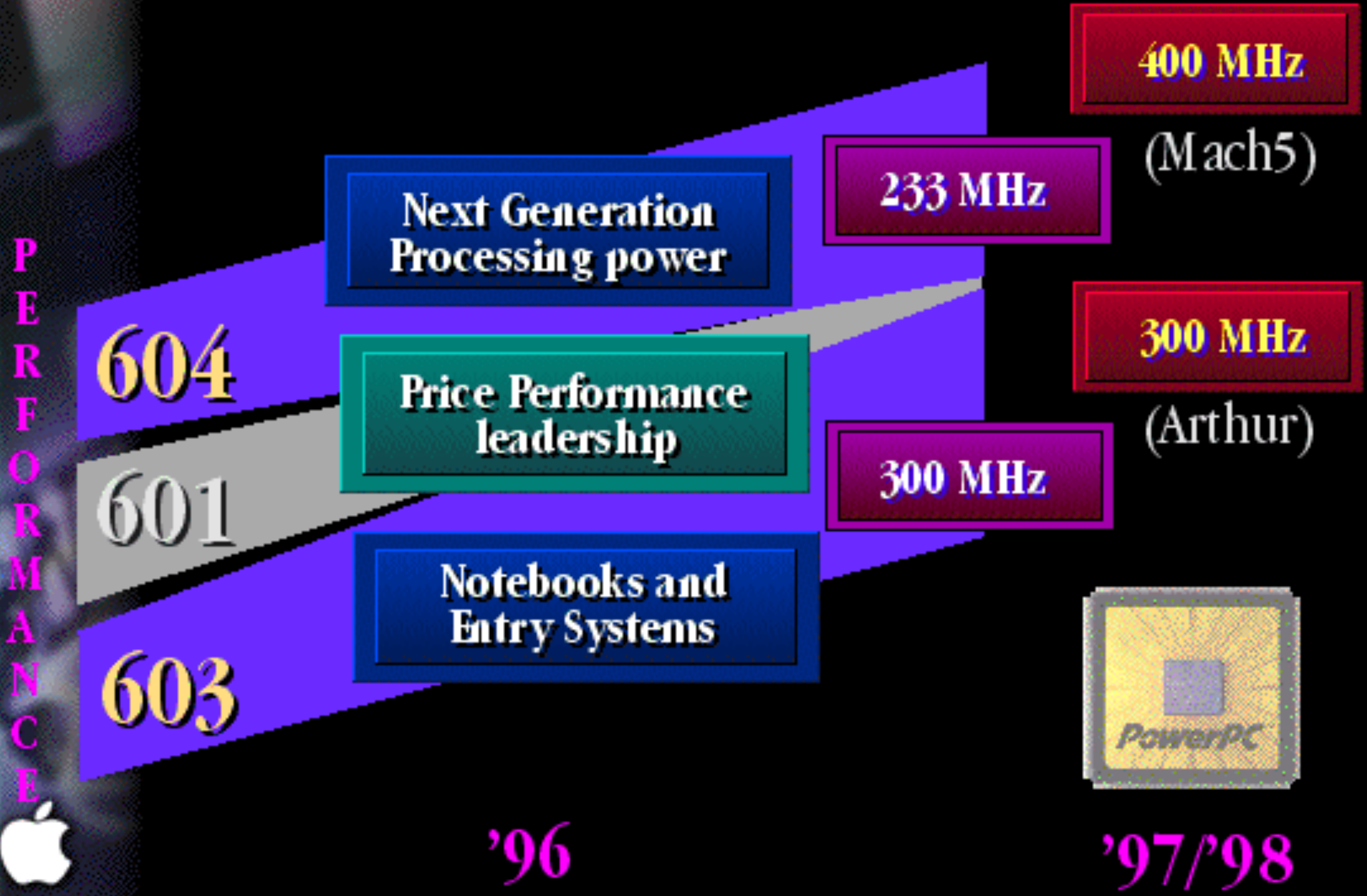
Kendall Luck

Power Mac
Product Marketing

One Year Later

- Shipping Hardware: Apple, DayStar, Power Computing, UMAX
- MP apps last year: Adobe, Deneba, Electric Image, Metrowerks, MetaTools, Strata and Specular
- New app support for Apple MP API: Lightworks, Orphan Technologies, Be Inc., Pixel, Terran Interactive, Vertigo, NewTek

1997 PowerPC Directions


PERFORMANCE
604
601
603
Next Generation Processing power
Price Performance leadership
Notebooks and Entry Systems
233 MHz
300 MHz
400 MHz
(Mach5)
300 MHz
(Arthur)
PowerPC
'96
'97/'98

MP Goals

- System 7
  - Continued support for the Apple MP API
- Rhapsody
  - Achieve SMP

How to Get More Information

- Apple HW Evangelist—David Masamitsu [mpevangelism@apple.com]

Multiprocessing Strategies for Mac OS and Rhapsody

Russell Williams

Senior Engineer

Still a Major Direction

- Parallelism at all levels: instruction, data, thread
- Threading especially relevant to 3D and multimedia
- MP: Proven safe and effective
- Coming to your customers' desktops
  - Processors will be free:
    - By 2000, ~100M transistors / chip
    - Today, 603e core is ~.6M transistors

One Model, One API, Two Implementations

- Asymmetric in Mac OS
- Symmetric in Rhapsody
- Runs on all Power Macintoshes, both UP and MP
- Compute-intensive threads in Mac OS
- Native Rhapsody apps get more powerful models

The Model

- Hardware is symmetric
  - CPUs are the same
  - Caches are coherent
- Memory is shared between threads
- Coarse-grained, compute-only threads
- MP tasks scheduled preemptively on each CPU
- No direct toolbox or OS calls
- Main thread must poll in event loop

The API

- 20 calls—two new ones since last year
- 2 concepts:
  - Tasks / threads (nomenclature clash)
  - Synchronization / communication
- MPRPC allows callback to main thread
- Supported in Mac OS, Rhapsody Blue Box

Tasks / Threads

- Scheduled preemptively on all processors
- Scheduling algorithms not specified
- On Rhapsody:
  MPTask == NSThread ==
  Java thread == cthread == Mach thread

Synchronization

- Never synchronize via scheduling (no safety in WaitNextEvent)
- Only single aligned scalar stores are inherently atomic
- Synchronization facilities:
  - Atomic operations (lockless)
  - Semaphores
  - Critical regions
  - Primitive messages

Mac OS Implementation

- Asymmetric OS, symmetric API
- VM not supported except on UP
- Limited preemption on main cpu
- Debugging via Mac OS-only routines and MW debugger
- Prerelease MP-safe stdclib on ETO #23

Rhapsody Implementation

- MP tasks become Mach threads
- SMP: any task or thread runs on any CPU
- Debugging via standard tools
- Mac OS—only debugging calls not supported

How to Use the MP API

- Create queues and synchronization objects
- Create MPProcessors()-1 tasks
- Communicate with the tasks
- Terminate the tasks

Thread Creation Example

```
err = CreateQueue(&requestQueue);
err = CreateQueue(&replyQueue);
for (i=0;i<MPProcessors()-1;i++)
  err = MPCreateTask(&MyTask,
    taskParam[i],
    kMPUseDefaultStackSize,
    replyQueue, nil, nil,
    kMPNormalTaskOptions,
    &taskID[i]);
```

Messaging Example

- Sender:
  - status = MPNotifyQueue(requestQueue, p1, p2, p3);
- Receiver:
  - status = MPWaitOnQueue(requestQueue, &p1, &p2, &p3, kDurationForever);

Using the Toolbox from an MP Task

- MPTaskIsToolboxSafe returns true if toolbox calls are OK
- MPRPC blocks until main thread calls WaitNextEvent
- At each WNE call, main thread empties MPRPC work queue
- void *MyToolboxUsingFunc(
    void *param);
- result = MPRPC(
    MyToolboxUsingFunc, param);

More Stuff in Native Rhapsody

- NSThread class supports OO thread model in Yellow Box
- NSThreads can call: stdclib subset, system calls, Foundation Kit, DPS, AppleEvents
- NSThreads cannot call App Kit
- cthreads can call stdclib subset, system calls, DPS
- Driver Kit drivers are MP-safe
- Java threads of course

MP Task Tips

- Correctness:
  - No 68K code
  - No preemption on main cpu when QuickTime is running (Mac OS)
  - No Mac OS toolbox calls (beware callbacks)
- Performance:
  - Substantial
  - Memory bandwidth
  - Avoid contention for globals
  - Watch out for lock contention

Worldwide
Developers
Conference