A Couple Music Applications

Nils Peters and Eric Battenberg
Parallel Computing Laboratory (Par Lab)
The Center for New Music and Audio Technologies (CNMAT)
University of California, Berkeley

4th Annual Short Course on Parallel Programming, August 15- 17, 2012

The 12 motifs

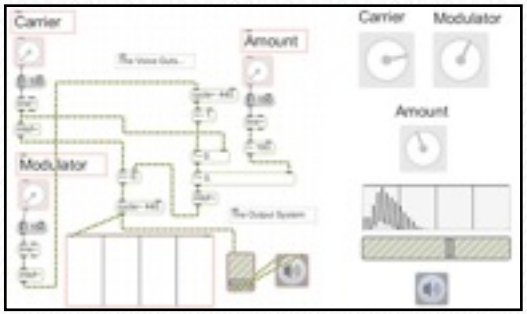
		Embed	SPEC	DB	Games	ML	CAD	HPC	Health	Image	Speech	Music	Browser
1	Finite State Mach.												
2	Circuits												
3	Graph Algorithms												
4	Structured Grid												
5	Dense Matrix												
6	Sparse Matrix												
7	Spectral (FFT)												
8	Dynamic Prog												
9	Particle Methods												
10	Backtrack/ B&B												
11	Graphical Models												
12	Unstructured Grid												

Range of Apps

Hundreds of apps and plug-ins

Music Information Retrieval / Music Recommendation

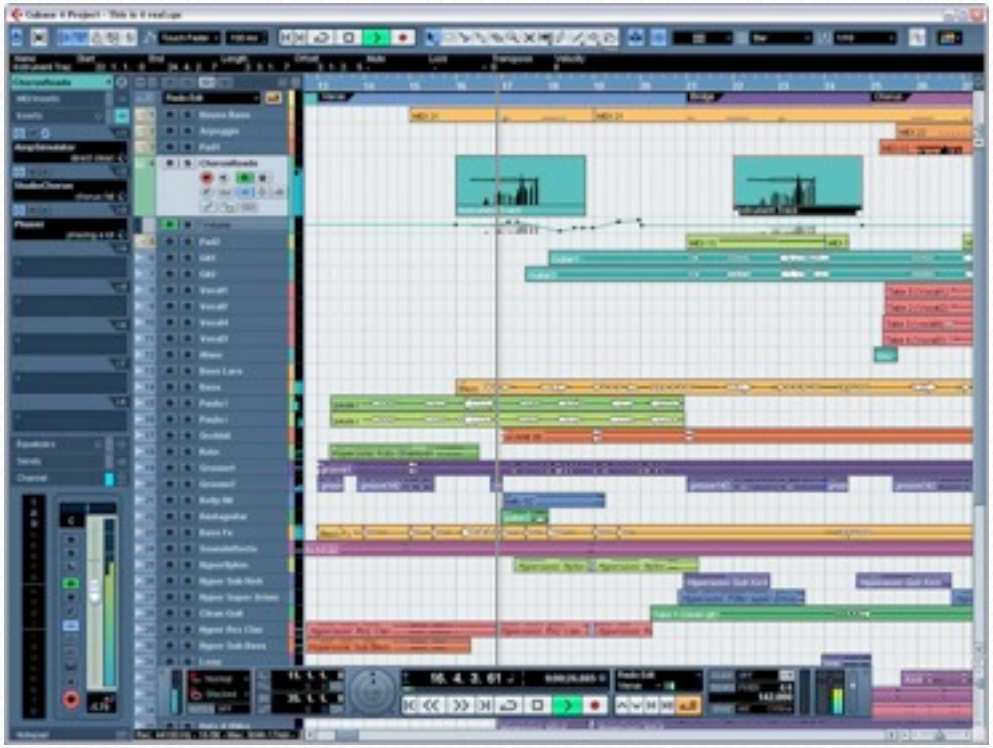
New Musical Instruments / Controller /Music Creation



3D Loudspeakers /Microphone Arrays

Voices, Streams, Channels, Tracks, and Lines

Music's Low Hanging Fruit Ripe for Parallelism



Usually a mix of task parallelism and data parallelism

Two music applications

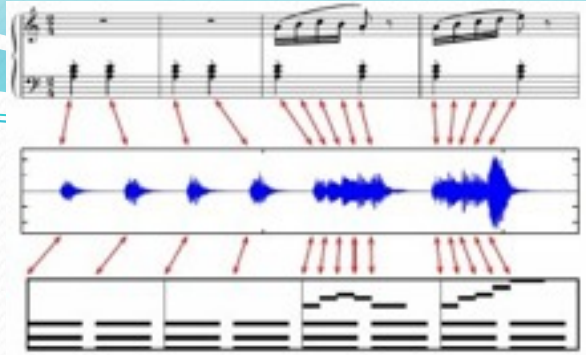
1. Music Information Retrieval (MIR)

   Drum track extraction using Open MP and CUDA

2. Real-time audio processing

   Partitioned convolution using Pthreads

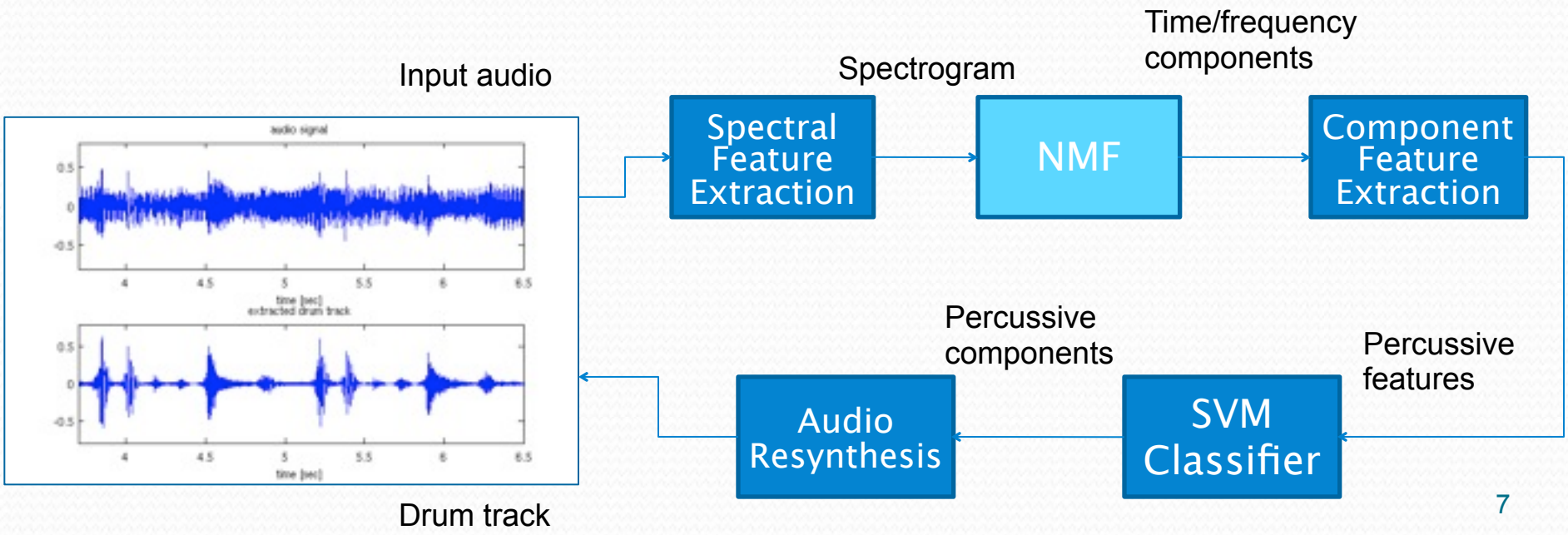
A few MIR tasks



- Music Information Retrieval, “Machine Listening”, “Music Understanding”
- **Transcription** – Automatically generate a score or tablature from audio
- **Source separation** – Isolate certain instruments (including vocals)
- **Song similarity, playlist creation, music recommendation**
- **Artist, genre, mood classification**
  - Help organize a music archive
- **Score Following, lyrics sync, beat tracking**
  - Useful for DJs, karaoke, music education, and automated accompaniment.
- **Song Segmentation**
  - Partition song into discrete passages (verse, chorus, bridge) The hope is that someday you will be able to query for music like this:

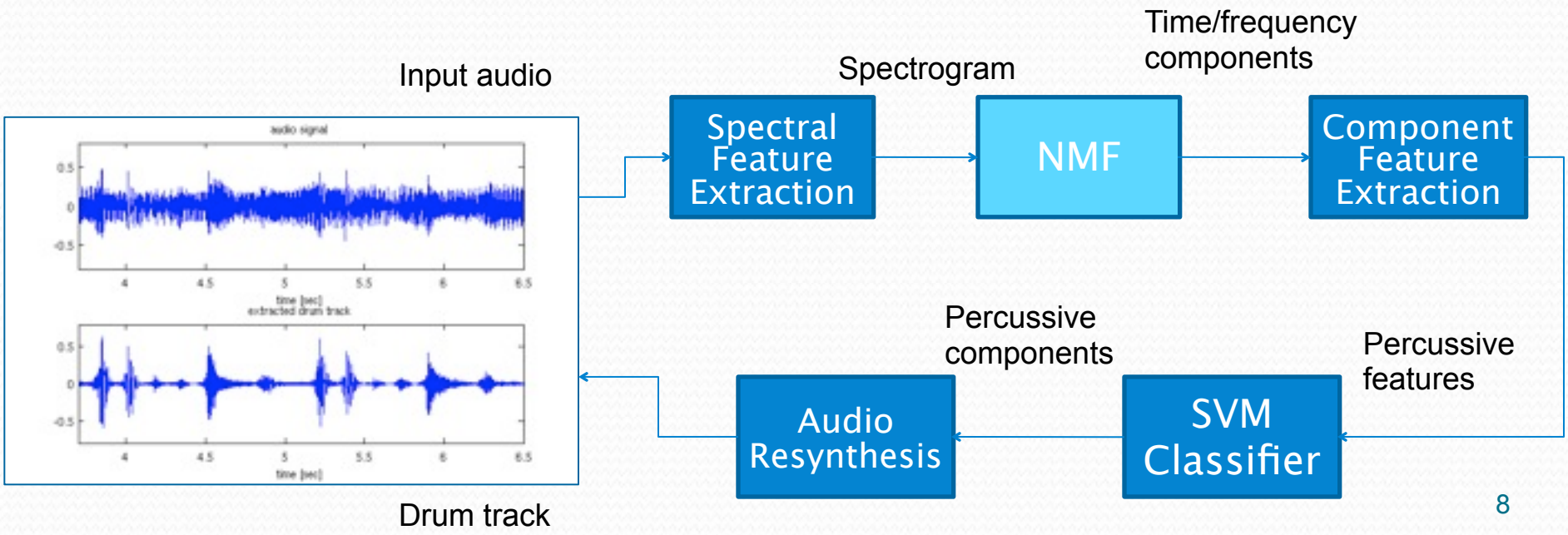
Drum Track Extraction

Audio examples (listen for drums in original)



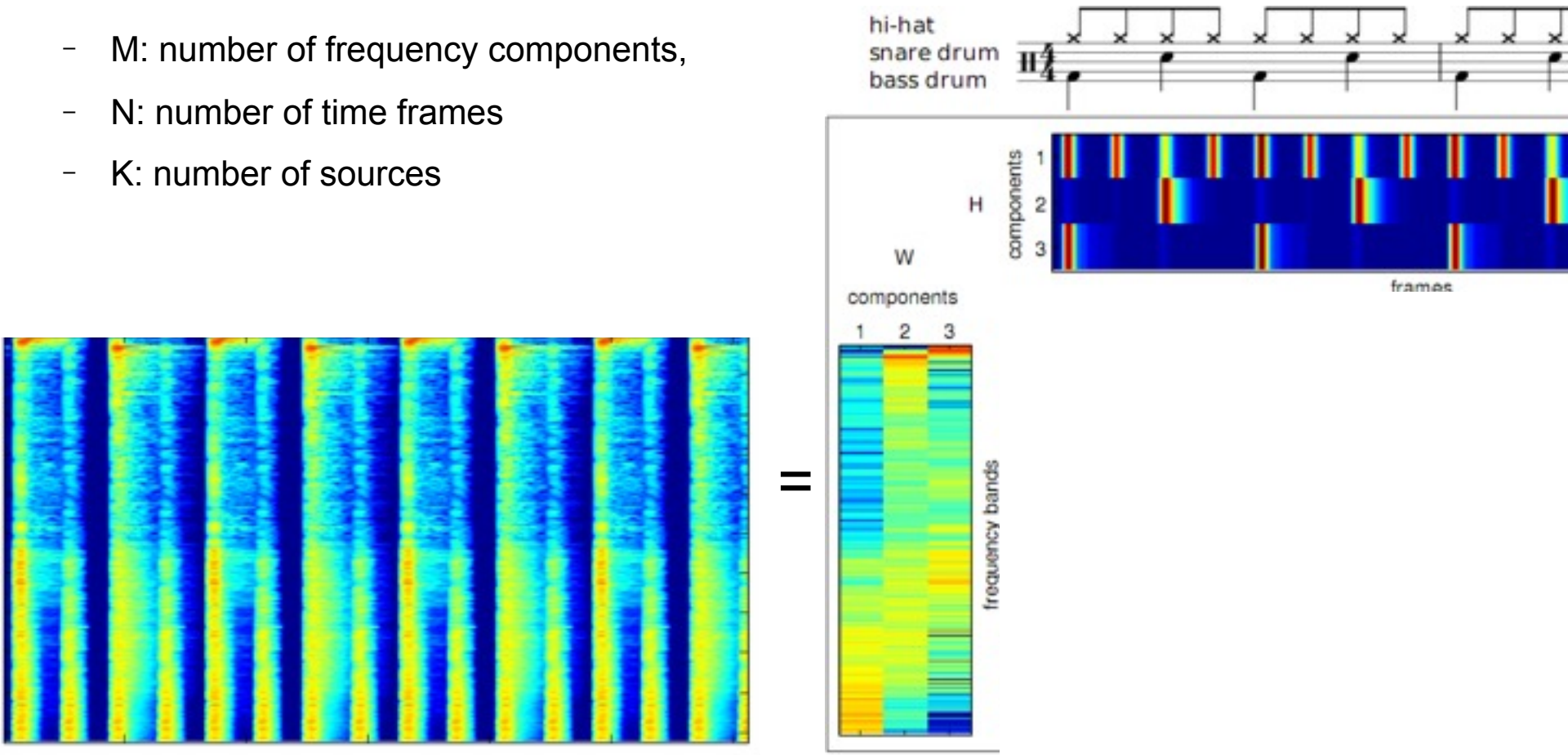
Drum Track Extraction

- An example of source separation where the drum track is isolated.
  - Useful in drum transcription, beat tracking, and rhythm analysis.
- Audio spectrogram is factorized into components using Non-negative Matrix Factorization (NMF).
- Components are classified using a Support Vector Machine (SVM).
- "Percussive" components are used to synthesize an audio drum track.
- **NMF step is most computationally intensive.**
  - 80% of time in Matlab (18.5 sec of 23.1 sec total for 20 sec of audio)
  - **We will parallelize NMF using OpenMP (for multi-core) and CUDA (for GPUs)**



Drum Track Extraction

- NMF source separation is accomplished using an iterative gradient-based solver
  - Our aim is to factorize a positive matrix X (MxN) into 2 positive matrices H (MxK) and W (KxN), where K is much smaller than M and N
  - M: number of frequency components,
  - N: number of time frames
  - K: number of sources



Drum Track Extraction

- For [512 x 30 x 3445] NMF,
  - 512 frequency components, 30 sources, 3445 time frames (~20 sec)
- For each iteration we have:
  - 423 Mflops of SGEMMs (**S**ingle-precision **Ge**neral **M**atrix **M**ultiply)
  - 3.6 Mflops of element-divides (slow)
  - 0.1 Mflops element-multiplies
  - 0.1 Mflops sums (requires communication)

$$\mathbf{H} \leftarrow \mathbf{H}.* \frac{\mathbf{W}^{\mathrm{T}} \frac{\mathbf{X}}{\mathbf{WH}}}{\mathbf{W}^{\mathrm{T}}\mathbf{1}}, \qquad \mathbf{W} \leftarrow \mathbf{W}.* \frac{\frac{\mathbf{X}}{\mathbf{WH}}\mathbf{H}^{\mathrm{T}}}{\mathbf{1}\mathbf{H}^{\mathrm{T}}}$$

- Also:
  - Add a small constant to divisor matrices to prevent divide-by-zero. (Add EPS, 3.6 Mflops)
  - Compute log-based cost function every 25 iterations to check for convergence.

$$D(\mathbf{X}\|\mathbf{WX}) = \sum_{ij}\left(\mathbf{X}_{ij}\log\frac{\mathbf{X}_{ij}}{(\mathbf{WH})_{ij}} - \mathbf{X}_{ij} + (\mathbf{WH})_{ij}\right)$$

NMF in OpenMP

- Data-parallel for loop
  - To be used for **element-wise arithmetic**
  - Create team of nt threads to do independent chunks of work

```
#pragma omp parallel for num_threads(nt)
    for(i=0;i<N;i++)
        c[i] = a[i]*b[i];
```

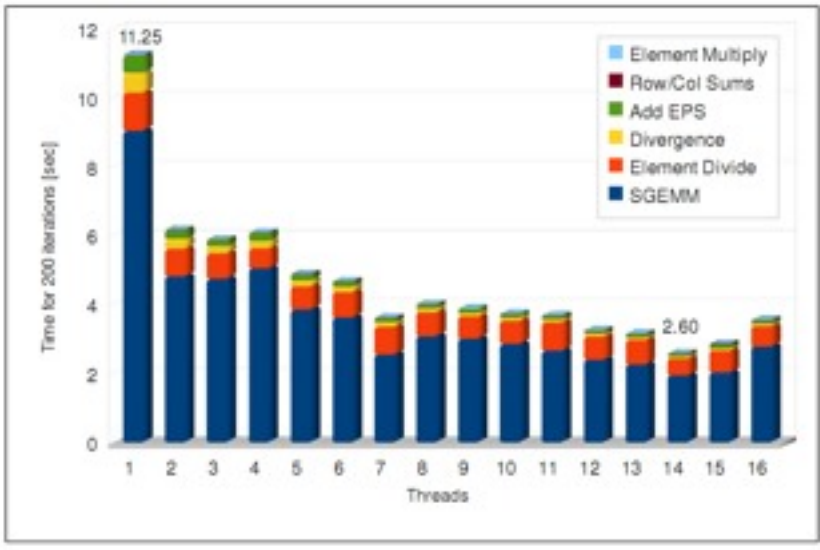
- Reduction
  - **For sums**
  - Create team of nt threads to compute partial sums
  - Then add the partial sums to final variable s

```
    s = 0;
#pragma omp parallel num_threads(nt)
#pragma omp for reduction(+:s)
    for(i=0;i<N;i++)
        s += a[i];
```

NMF in OpenMP

- We use MKL for SGEMMs
- Use OpenMP for other routines
- Performance scaling on dual-socket Core i7 920:

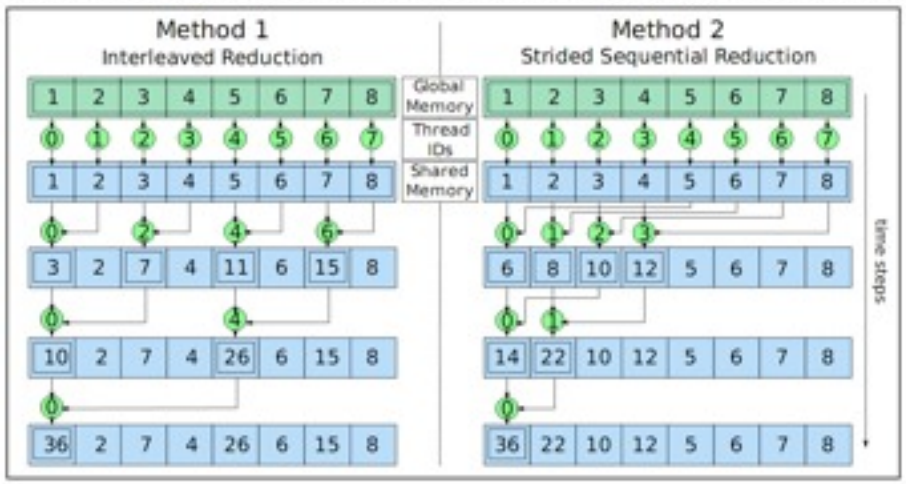
- SGEMMs show most significant speedup
  - Highest work to communication ratio
- Non-linear speedup suggests this won't scale well to more cores using this architecture and programming model.
- However,
  - >7x speedup compared to Matlab
  - >4x speedup compared to sequential C

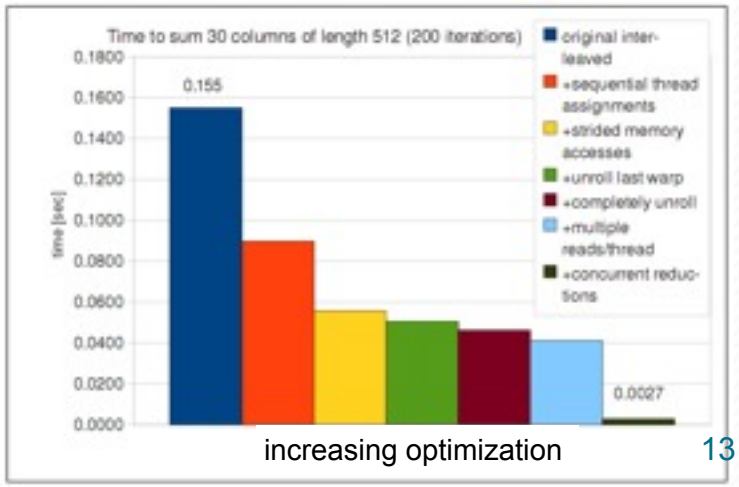


NMF in CUDA

- NMF Implementation in CUDA
  - SGEMMs – use CUBLAS 2.1, achieves 60% of peak (373 GFLOPS on GTX 280)
    - Padding matrices to multiples of 32 reduces SGEMM running time by 26%
  - Element-wise arithmetic – similar to example code
  - Reductions (sums) – a lot harder in CUDA than OpenMP
    - Use optimizations covered in CUDA SDK for shared memory reduction.
      - Reorganize binary tree traversal.
      - Loop unrolling, multiple reads per thread.
    - Run the 30 sums concurrently. An important optimization.

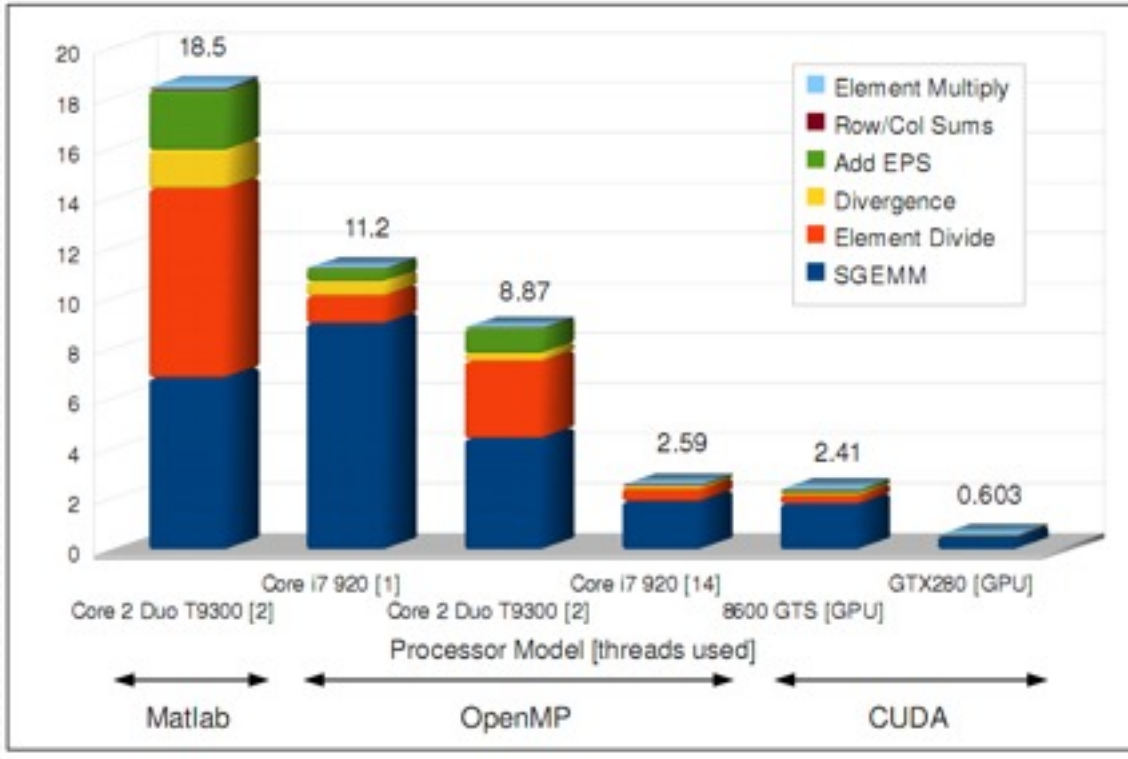
57x speedup overall





CUDA vs. OpenMP

- CUDA achieves much higher performance on current GPUs for highly data-parallel computations. (>30x speedup compared to Matlab, 4x faster than OpenMP+Nehalem)
- OpenMP can achieve multi-core speedup on data-parallel computations with very little programmer effort.
- If inter-thread communication is required, things become much more difficult.
  - OpenMP gets harder.
  - CUDA gets a lot harder.
- For music application developers, CUDA is only feasible for computational kernels that require very high performance.



Real-Time Audio Processing

- Very different requirements compared to offline batch MIR.
- Low-latency processing requires sub-10ms latency.
- If processing deadlines are missed, we hear a click in the audio stream
  - This must be avoided!
- Aim is to minimize the worst-case execution time of processing task.

User interfaces with pressure-sensitive multi-touch gestural interfaces

Convolution Reverb

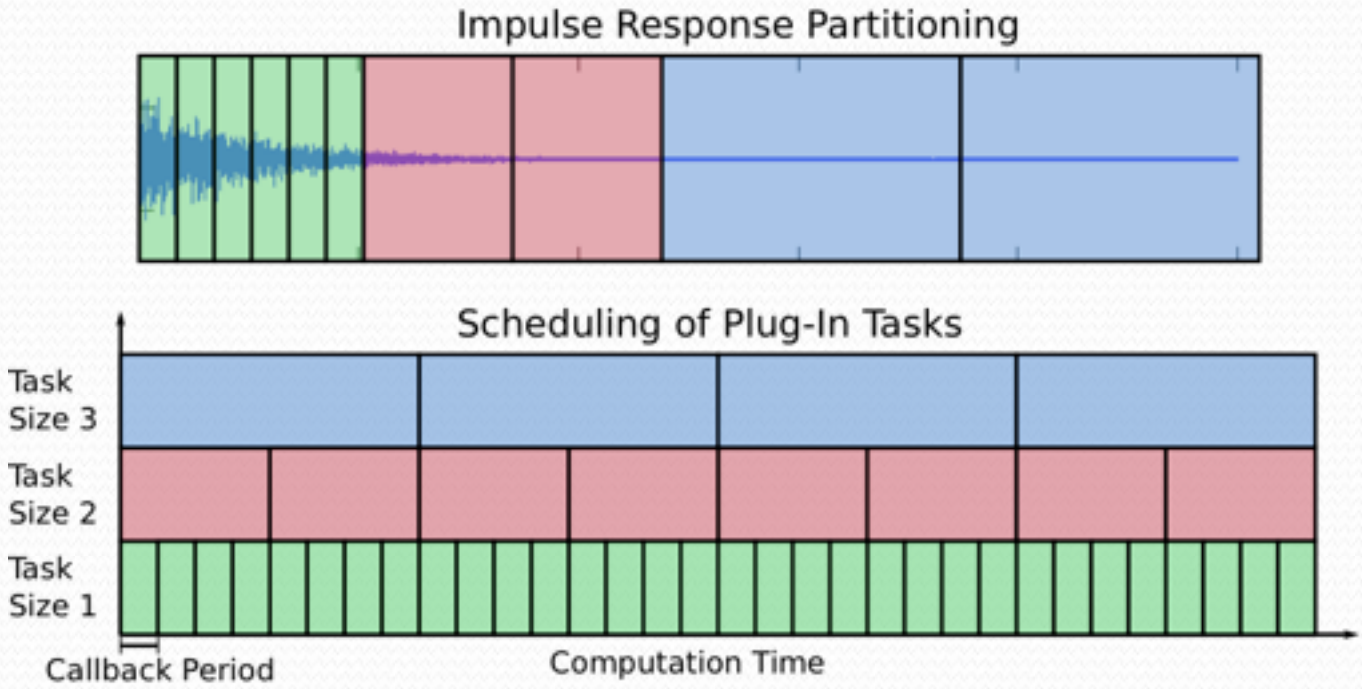
$$x(t) \quad * \quad h(t) \quad = \quad y(t)$$

Non-Uniform Partitioned Convolution

- Very important technique for:
  - Efficient, low-latency, real-time convolution reverb and FIR filtering.
- Shorter partitions -> lower latency
- Longer partitions -> higher computational efficiency

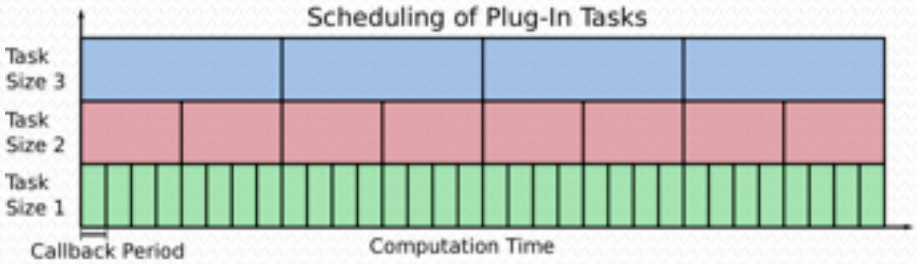
Scheduling Partitioned Convolution

- The processing of larger partitions takes longer but occurs less frequently.
- How do we schedule these long-running tasks?

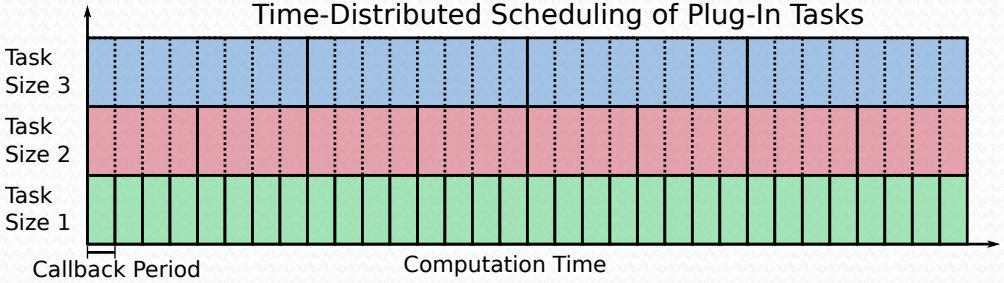


(Better) Scheduling Options for Long-Running Tasks (LRTs)

- 3. **Preempt** LRTs with higher-priority tasks when necessary.

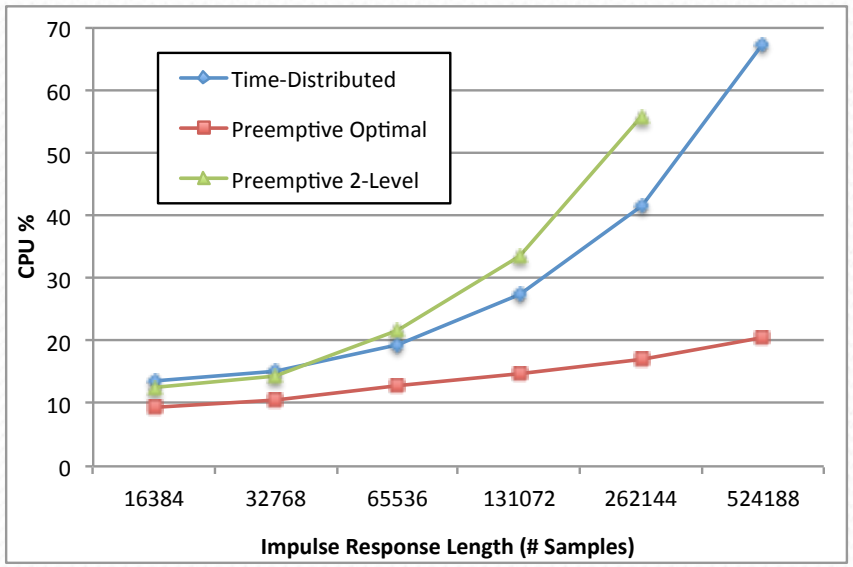


- 4. **Time-distribute** LRTs by breaking into sub-tasks.

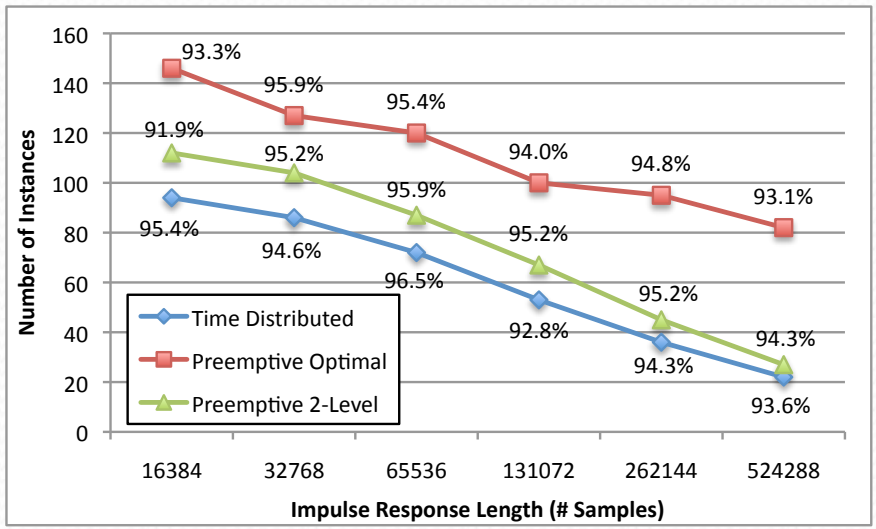


Scheduling Performance Results

- DAFx '11 Paper: Implementing Real-Time Partitioned Convolution Algorithms on Conventional Operating Systems. (Battenberg, Avizienis).
- Preemption vs. Time-Distributed (Single core)
- Preemptive version:
  - Better performance
  - Easier to program



CPU Load



Reliability

Summary

- Music apps are valuable use cases for exploiting parallelism
- Data-parallelism and task-parallelism

- Two examples for parallel computing in music applications
- 1. Music Information Retrieval (MIR)
- 2. Real-Time Audio Processing

- Different ways of implementations based on requirements

A Couple Music Applications

Nils Peters and Eric Battenberg
Parallel Computing Laboratory (Par Lab)
The Center for New Music and Audio Technologies (CNMAT)
University of California, Berkeley

{nils, ericb}@eecs.berkeley.edu

4th Annual Short Course on Parallel Programming, August 15- 17, 2012

References

- NMF on Open MP and CUDA:
  - E. Battenberg and D. Wessel, “Accelerating non-negative matrix factorization for audio source separation on multi-core and many-core architectures” ,ISMIR, 2009.
- Real-Time Partitioned Convolution:
  - E. Battenberg and R. Avizienis, “Implementing Real-Time Partitioned Convolution Algorithms on Conventional Operating Systems,” DAFX, 2011.
  - J. Colmenares, E. Battenberg, N. Peters, I. Saxton, R. Avizienis, J. Kubiatowicz, and D. Wessel. Real-time music applications on an experimental operating system for many cores. ICMC, 2011.