Antisocial Parallelism: Avoiding, Hiding and Managing Communication

Kathy Yelick
Associate Laboratory Director of Computing Sciences
Lawrence Berkeley National Laboratory

EECS Professor, UC Berkeley
“Big Data” Changes Everything…What about Science?
Transforming Science: Finding Data

Computing Challenges:
• Search for scientific data on the web
• Automated metadata annotation / feature identification
• Data: images, genomes, simulations, MRI, MassSpec,…
Scientific Workflow Today

Experiment

Beamline User
The Future of Experimental Science

Simulation & Analysis Framework

Beamline
User

Data Pipeline

HPC Storage & Compute

New experiment

Experiments

Prompts

Comparison

Simulation & Analysis Framework

Science Gateway

Sure
Transforming experimental science: “Superfacility” for Science

Computing Challenges:
- Robotics, Special purpose processors at experiments
- Mathematics / algorithm for real-time and offline analysis
- Massive numbers of simulations for inverse problems
- Networks and software for data movement, management
Science at the Boundary of Simulation and Observation

Adaptive Mesh Refinement simulates sea level impacts from melting of West Antarctic Ice Sheet

Deep learning algorithms identify and help quantify extreme events

Computing Challenges:

- Multimodal analysis from sensors, genomes, images...
- High performance methods and implementations
- Data-driven simulations to predict regional effects on environment and weather events
Finding smaller signals in noisy, biased data: Removing Systematic Bias in Cosmology

Example: Astrophysicists discover early nearby supernova

Computing Challenges:
- Better machine learning for event detection
- Removing systematic bias in experimental data
- Simulations to interpret data; data constrain simulations
Finding structure and function in noisy data: Metagenomics data mining

Computing Challenges:
- Distributed memory graph algorithms / hash tables
- Low latency interconnects; low overhead communication
- Algorithms to separate and assembly genomes
- Many-to-Many comparisons against databases
Science Trends

- Science needs (and will always need) more computing
- New science questions at the boundary of simulation and observation
- Changes to computing infrastructure needed for open, reproducible science
Roadmap

✔ Science Trends
• Political Trends
• Technology Trends
• Algorithmic Challenges
The Politics of High Performance Computing
White House Announces the National Strategic Computing Initiative (NSCI)

For Immediate Release  
July 29, 2015

EXECUTIVE ORDER

CREATING A NATIONAL STRATEGIC COMPUTING INITIATIVE

By the authority vested in me as President by the Constitution and the laws of the United States of America, and to maximize benefits of high-performance computing (HPC) research, development, and deployment, it is hereby ordered as follows:

Section 1. Policy. In order to maximize the benefits of HPC for economic competitiveness and scientific discovery, the United States Government must create a coordinated Federal strategy in HPC research, development, and deployment. Investment in HPC has contributed substantially to national economic prosperity and rapidly accelerated scientific discovery. Creating and deploying technology at the leading edge is vital to advancing my Administration’s priorities and spurring innovation. Accordingly, this order establishes the National Strategic Computing Initiative (NSCI). The NSCI is a

Five goals:

1. Create systems that can apply exaflops of computing power to exabytes of data.
2. Keep the United States at the forefront of HPC capabilities.
3. Improve HPC application developer productivity.
4. Make HPC readily available.
5. Establish hardware technology for future HPC systems.

[DOE SC and NNSA] will execute a joint program focused on advanced simulation through a capable exascale computing ...
Advanced Computing: Not just for Simulation

- Experimentation
- Theory
- Data Analysis
- Simulation

Comprehensive Test ban treaty

Petascale Computing for Small Number of Hero Simulations
Science Needs Computing for Both Experiments (Data) and Theory (Modeling and Simulation)

Commercial “Big Data”
Growth in Sequencers, CCDs, etc.

Computing foundation includes research (math/stat and CS) and facilities (data and compute)

Future Performance from Exascale Technology
The Project has three phases:
- Phase 1 – R&D before DOE facilities exascale systems RFP in 2019
- Phase 2 – Exascale architectures and NRE are known. Targeted development
- Phase 3 – Exascale systems delivered. Meet Mission Challenges
Proposed DOE Exascale Science Problems

<table>
<thead>
<tr>
<th>#</th>
<th>Site</th>
<th>Manufacturer</th>
<th>Computer</th>
<th>Country</th>
<th>Cores</th>
<th>Rmax [Pflops]</th>
<th>Power [MW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>National Supercomputing Center in Wuxi</td>
<td>NRCPC</td>
<td>Sunway TaihuLight</td>
<td>China</td>
<td>10,649,600</td>
<td>93.0</td>
<td>15.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NRCPC Sunway SW26010,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>260C 1.45GHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>National University of Defense Technology</td>
<td>NUDT</td>
<td>Tianhe-2</td>
<td>China</td>
<td>3,120,000</td>
<td>33.9</td>
<td>17.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NUDT TH-IVB-FEP,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Xeon 12C 2.2GHz,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IntelXeon Phi</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Oak Ridge National Laboratory</td>
<td>Cray</td>
<td>Titan</td>
<td>USA</td>
<td>560,640</td>
<td>17.6</td>
<td>8.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XK7,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Opteron 16C 2.2GHz,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gemini, NVIDIAXeon Phi</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Lawrence Livermore National Laboratory</td>
<td>IBM</td>
<td>Sequoia</td>
<td>USA</td>
<td>1,572,864</td>
<td>17.2</td>
<td>7.89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BlueGene/Q,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Power BQC 16C 1.6GHz,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Custom</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>RIKEN Advanced Institute for Computational Science</td>
<td>Fujitsu</td>
<td>K Computer</td>
<td>Japan</td>
<td>795,024</td>
<td>10.5</td>
<td>12.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SPARC64 VIIIfx 2.0GHz,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tofu Interconnect</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Argonne National Laboratory</td>
<td>IBM</td>
<td>Mira</td>
<td>USA</td>
<td>786,432</td>
<td>8.59</td>
<td>3.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BlueGene/Q,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Power BQC 16C 1.6GHz,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Custom</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Los Alamos NL / Sandia NL</td>
<td>Cray</td>
<td>Trinity</td>
<td>USA</td>
<td>301,0564</td>
<td>8.10</td>
<td>4.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XC40,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Xeon E5 16C 2.3GHz,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Aries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Swiss National Supercomputing Centre (CSCS)</td>
<td>Cray</td>
<td>Piz Daint</td>
<td>Switzerland</td>
<td>115,984</td>
<td>6.27</td>
<td>2.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XC30,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Xeon E5 8C 2.6GHz,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Aries, NVIDIAXeon E5 K20x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>HLRS – Stuttgart</td>
<td>Cray</td>
<td>Hazel Hen</td>
<td>Germany</td>
<td>185,088</td>
<td>5.64</td>
<td>3.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XC40,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Xeon E5 12C 2.5GHz,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Aries</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>King Abdullah University of Science and Technology</td>
<td>Cray</td>
<td>Shaheen II</td>
<td>Saudi Arabia</td>
<td>196,608</td>
<td>5.54</td>
<td>2.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cray XC40,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Xeon E5 16C 2.3GHz,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sunway TaihuLight

- 125.4 Pflop/s theoretical peak
- SW26010 processor, 1.45 GHz
- Node = 260 Cores (1 socket)
  - 4 – core groups; 32 GB memory (DDR3)
- 40,960 nodes in the system
  - 10,649,600 cores total
- 1.31 PB of primary memory
- 93 Pflop/s HPL, 74% peak
- 15.3 Mwatts (6 MF/Watt)

One piece of entire computing strategy on applications, fabs, architecture, software
Roadmap

✔ Science Trends
✔ Political Trends
  • Technology Trends
  • Algorithmic Challenges
Technology Trends
Computing is energy-constrained

At ~$1M per MW, energy costs are substantial

- 1 petaflop in 2008 used 3 MW
- 1 exaflop in 2018 at 200 MW “usual chip scaling”

Goal: 1 Exaflop in 20 MW
= 20 pJ / operation

Note: The 20 pJ / operation is
- Independent of machine size
- Independent of # cores used per application
- But “operations” need to be useful ones
Multi-Core is NOT good enough! *(need to go to simpler cores)*

Can continue with conventional x86 architectures if you want.

Lightweight cores OR Hybrid is the only approach that crosses the exascale finish line
Communication Consumes Energy

Latency is physics; bandwidth is money, … but overhead we can fix
Roadmap

✔️ Science Trends
✔️ Political Trends
✔️ Technology Trends
• Algorithmic Challenges
Algorithm Challenge: Communication
### Analytics vs. Simulation Kernels:

<table>
<thead>
<tr>
<th>7 Giants of Data</th>
<th>7 Dwarfs of Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic statistics</td>
<td>Monte Carlo methods</td>
</tr>
<tr>
<td>Generalized N-Body</td>
<td>Particle methods</td>
</tr>
<tr>
<td>Graph-theory</td>
<td>Unstructured meshes</td>
</tr>
<tr>
<td>Linear algebra</td>
<td>Dense Linear Algebra</td>
</tr>
<tr>
<td>Optimizations</td>
<td>Sparse Linear Algebra</td>
</tr>
<tr>
<td>Integrations</td>
<td>Spectral methods</td>
</tr>
<tr>
<td>Alignment</td>
<td>Structured Meshes</td>
</tr>
</tbody>
</table>

There are some differences between data and simulation algorithms, but more similarities than differences. Some of the data algorithms use no arithmetic (genomics) or lower precision (deep learning).
Never Waste Fast Memory

Don’t get hung up on the “owner computes” rule.
Beyond Domain Decomposition: 2.5D Matrix Multiply

- Conventional “2D algorithms” use $P^{1/2} \times P^{1/2}$ mesh and minimal memory
- New “2.5D algorithms” use $(P/c)^{1/2} \times (P/c)^{1/2} \times c^{1/2}$ mesh and $c$-fold memory

**Surprises:**
- Even Matrix Multiply had room for improvement
- Idea: make copies of C matrix (as in prior 3D algorithm, but not as many)
- Result is provably optimal in communication

**Lesson:** Never waste fast memory

Can we generalize for compiler writers?

![Graph](image-url)
Deconstructing 2.5D Matrix Multiply
Solomonick & Demmel

- Tiling the iteration space
- 2D algorithm: never chop k dim
- 2.5 or 3D: Assume + is associative; chop k, which is → replication of C matrix

Matrix Multiplication code has a 3D iteration space
Each point in the space is a constant computation (*/+)

for i
  for j
    for k
      C[i,j] ... A[i,k] ... B[k,j] ...
Lower Bound Idea on $C = A*B$

Iromy, Toledo, Tiskin

Cubes in black box with side lengths x, y and z

= Volume of black box

= x*y*z

= ($\#A\Box s \times \#B\Box s \times \#C\Box s$)$^{1/2}$

= (xz * zy * yx)$^{1/2}$

(i,k) is in “A shadow” if (i,j,k) in 3D set

(j,k) is in “B shadow” if (i,j,k) in 3D set

(i,j) is in “C shadow” if (i,j,k) in 3D set

Thm (Loomis & Whitney, 1949)

# cubes in 3D set = Volume of 3D set

≤ (area(A shadow) * area(B shadow) * area(C shadow))$^{1/2}$
Generalizing Communication Lower Bounds and Optimal Algorithms

• For serial matmul, we know \#words\_moved = \( \Omega \left( \frac{n^3}{M^{1/2}} \right) \), attained by tile sizes \( M^{1/2} \times M^{1/2} \)

• Thm (Christ,Demmel,Knight,Scanlon,Yelick): For any program that “smells like” nested loops, accessing arrays with subscripts that are linear functions of the loop indices

\[
\#\text{words\_moved} = \Omega \left( \frac{\#\text{iterations}}{M^e} \right)
\]

for some \( e \) we can determine

• Thm (C/D/K/S/Y): Under some assumptions, we can determine the optimal tiles sizes
  – E.g., index expressions are just subsets of indices

• Long term goal: All compilers should generate communication optimal code from nested loops
Implications for Arithmetic

- Much of the work on compilers is based on owner-computes
  - For MM: Divide C into chunks, schedule movement of A/B
  - Data-driven domain decomposition partitions data; but we can partition work instead

- Ways to compute C “pencil”
  1. Serially
  2. Parallel reduction *Standard vectorization trick*
  3. Parallel asynchronous (atomic) updates
  4. Or any hybrid of these

- For what types / operators does this work?
  - “+” is associative for 1,2 rest of RHS is “simple”
  - and commutative for 3

Using $x$ for $C[i,j]$ here
Traditional (Naïve $n^2$) Nbody Algorithm (using a 1D decomposition)

- Given $n$ particles and $p$ processors, size $M$ memory
- Each processor has $n/p$ particles
- Algorithm: shift copy of particles to the left $p$ times, calculating all pairwise forces
- Computation cost: $n^2/p$
- Communication cost: $O(p)$ messages, $O(n)$ words
Communication Avoiding Version
(using a “1.5D” decomposition)

• Divide $p$ into $c$ groups. Replicate particles within group.
  – First row responsible for updating all by orange, second all by green, …

• Algorithm: shift copy of $n/(p\times c)$ particles to the left
  – Combine with previous data before passing further level (log steps)

• Reduce across $c$ to produce final value for each particle

• Total Computation: $O(n^2/p)$;  
  \[ \text{Limit: } c \leq p^{1/2} \]

• Total Communication: $O(\log(p/c) + \log c)$ messages,  
  \[ O(n^*(c/p+1/c)) \text{ words} \]
Challenge: Symmetry & Load Balance

- Force symmetry ($f_{ij} = -f_{ji}$) saves computation
- 2-body force matrix vs 3-body force cube

- How to divide work equally?

Koanantakool & Yelick
3-Way N-Body Animation

- $p=5$, $n=30$
- 6 particles per processor
- 5x5 subcubes

Koanantakool & Yelick
3-Way N-Body Animation

- \( p=5, \ n=30 \)
- 6 particles per processor
- 5x5 subcubes

Equivalent triplets in the big tetrahedron

Actual triplets
3-Way N-Body Animation

- $p=5$, $n=30$
- 6 particles per processor
- 5x5 subcubes

Actual triplets

Equivalent triplets in the big tetrahedron

Koanantakool & Yelick
3-Way N-Body Animation

- $p=5$, $n=30$
- 6 particles per processor
- 5x5 subcubes

**Koanantakool & Yelick**

**Actual triplets**

**Equivalent triplets in the big tetrahedron**
3-Way N-Body Speedup

- Cray XC30, 24k cores, 24k particles

Down is good

Koanantakool & Yelick
Strong Scaling of .5D Algorithms

Relative Efficiency vs. One Core

Machine size (# cores)

Koanantakool & Yelick
Sparse-Dense Matrix Multiply Too!

- Variety of algorithms that divide in or 2 dimensions

Koanantakool & Yelick
Have We Seen this Idea Before?

• These algorithms also maximize parallelism beyond “domain decomposition”
  – SIMD machine days
• Automation depends on associative operator for updates (e.g., M. Wolfe)
• Also used for “synchronization avoidance” in Particle-in-Cell code (Madduri, Su, Oliker, Yelick)
  – Replicate and reduce optimization given p copies
  – Useful on vectors / GPUs
Avoid Latency and Implicit Synchronization in Communication

- Two-sided message passing (e.g., send/receive in MPI) requires matching a send with a receive to identify memory address to put data
  - Couples data transfer with synchronization, which is sometimes what you want

- Using global address space decouples synchronization
  - Pay for what you need!
Avoid Synchronization from Applications

Computations as DAGs

View parallel executions as the directed acyclic graph of the computation

Cholesky 4 x 4

QR 4 x 4

Slide source: Jack Dongarra
Event Driven LU in UPC

- Assignment of work is static; schedule is dynamic
- Ordering needs to be imposed on the schedule
  - Critical path operation: Panel Factorization
- General issue: dynamic scheduling in partitioned memory
  - Can deadlock in memory allocation
  - “memory constrained” lookahead

![Diagram of event-driven LU in UPC]

Some edges omitted.
Sparse Cholesky

- Timings on NERSC Edison, 24 Intel Ivy-bridge cores per node

Running times for pwtk

- Fan-both algorithm by Jacquelin & Ng, in UPC++
OpenMP Loop Parallelism is the Wrong Level

- OpenMP is popular for its convenient loop parallelism
- Loop level parallelism is too coarse and too fine:
  - Too coarse: Implicit synchronization between loops limits parallelism and adds overhead
  - Too fine: Need to create larger chunks of serial work by combining across loops (fusion) to minimize data movement

```c
!$OMP PARALLEL DO
  DO I=2,N
    B(I) = (A(I) + A(I-1)) / 2.0
  ENDDO
!$OMP END PARALLEL DO
```
Sources of Unnecessary Synchronization

Loop Parallelism

```
!$OMP PARALLEL DO
   DO I=2,N
      B(I) = (A(I) + A(I-1)) / 2.0
   ENDDO
!$OMP END PARALLEL DO
```

“Simple” OpenMP parallelism implicitly synchronized between loops

Abstraction

Bulk Synchrononous

Less Synchronous

Accelerator Offload

```
!$acc data copyin(cix,ci1,ci2,ci3,ci4,ci5,ci6,ci7,ci8,ci9,ci10,ci11,
!$acc & ci12,ci13,ci14,r,b,uxyz,cell,rho,grad,index_max,index,
!$acc & ciy,ciz,wet,np,streaming_shuf1, &
!$acc & streaming_shuf1f,streaming_shuf2,streaming_shuf4,streaming_shuf5, &
!$acc & streaming_shuf7f,streaming_shuf8f,streaming_shuf9f,streaming_shuf10f, &
!$acc & streaming_shuf11n,streaming_shuf12n,streaming_shuf13n,streaming_shuf14n, &
!$acc & streaming_shuf1f,streaming_shuf2f,streaming_shuf4f,streaming_shuf5f, &
!$acc & streaming_shuf7f,streaming_shuf8f,streaming_shuf9f,streaming_shuf10f, &
!$acc & streaming_rbuf1f,streaming_rbuf2f,streaming_rbuf4f,streaming_rbuf5f, &
!$acc & streaming_rbuf7n,streaming_rbuf8n,streaming_rbuf9n,streaming_rbuf10n, &
!$acc & streaming_rbuf1f,streaming_rbuf2f,streaming_rbuf4f,streaming_rbuf5f, &
!$acc & streaming_rbuf7f,streaming_rbuf8f,streaming_rbuf9f,streaming_rbuf10f, &
!$acc & streaming_rbuf1f,streaming_rbuf2f,streaming_rbuf4f,streaming_rbuf5f, &
!$acc & streaming_rbuf7f,streaming_rbuf8f,streaming_rbuf9f,streaming_rbuf10f, &
!$acc & send_e,send_w,send_n,send_s,recv_e,recv_w,recv_n,recv_s)
do ii=1,ntimes
   call set_boundary_macro_press2
   call set_boundary_micro_press
   call collisiona
   call collisionb
   call recolor
```

Libraries

<table>
<thead>
<tr>
<th>Analysis</th>
<th>% barriers</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto</td>
<td>42%</td>
<td>13%</td>
</tr>
<tr>
<td>Guided</td>
<td>63%</td>
<td>14%</td>
</tr>
</tbody>
</table>

NWChem: most of barriers are unnecessary (Corvette)

The transfer between host and GPU can be slow and cumbersome, and may (if not careful) get synchronized
Thanks to many Collaborators!

- Jim Demmel
- Dan Bonachea
- Bill Carlson
- Michael Driscoll
- Marquita Ellis
- Evangelos Georganas
- Paul Hargrove
- Costin Iancu
- Khaled Ibrahim
- Shoaib Kamil
- Amir Kamil
- Penporn Kaonantakool
- Nick Knight
- Leonid Oliker
- Eric Roman
- Hongzhang Shan
- Edgar Solomonik
- Erich Strohmaier
- Rich Vuduc
- Sam Williams
- Yili Zheng
- Greg Balls
- Christian Bell
- Soumen Chakrabarti
- Wei Chen
- Kaushik Datta
- Etienne Deprit
- Jason Duell
- Tarek El-Ghazawi
- Ed Givelberg
- Parry Husbands
- Eun-Jin Im
- Jeff Jones
- Brian Kazian
- Arvind Krishnamurthy
- C.J. Lin
- Sabrina Merchant
- Carelton Miyamoto
- Mani Narayanan
- Rajesh Nishtala Steve
- Steinberg Jimmy Su
- Randi Thomas
- Noah Treuhaft
- Chih-Po Wen
- Siu-Man Yau