



Using ClearSpeed Accelerators in Parallel from Very High-Level Languages

Steve Reinhardt, VP of Joint Research



INTERACTIVE
supercomputing

Agenda

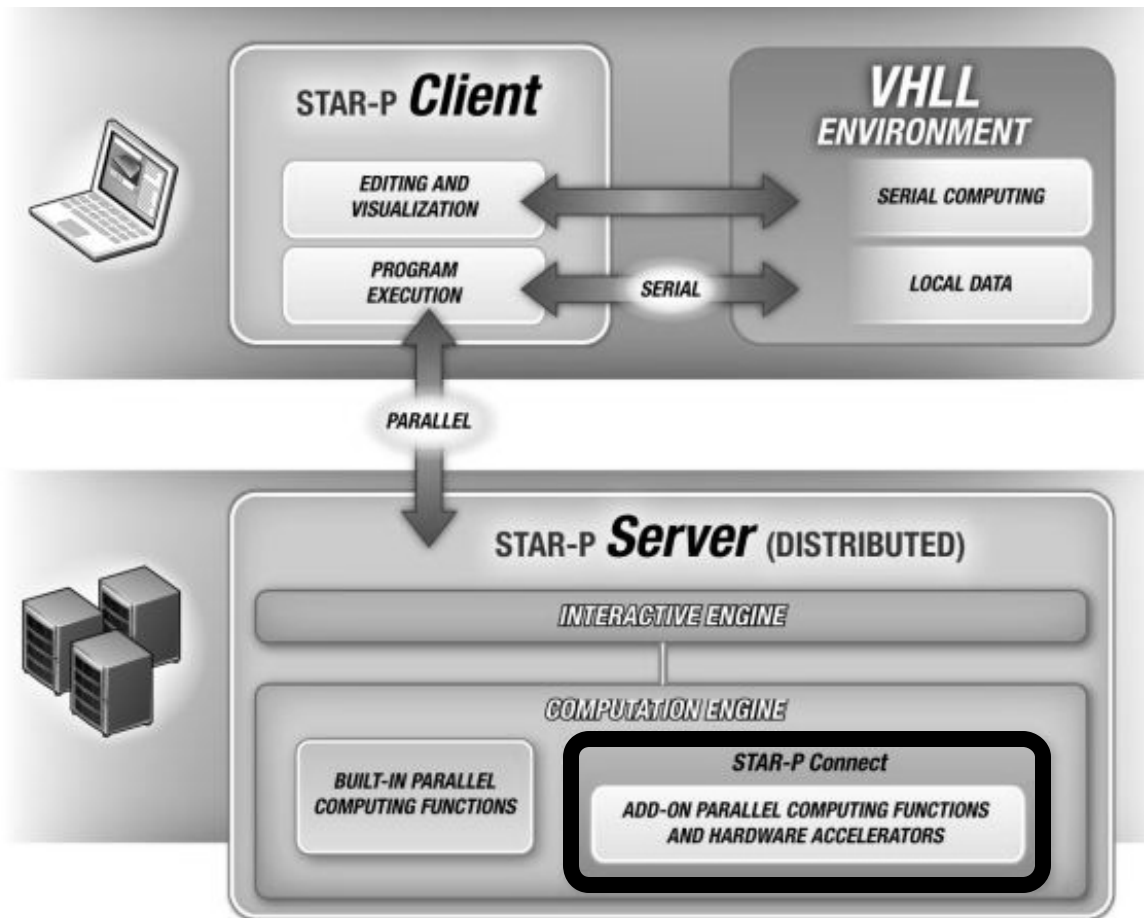
- Star-P basics
- Using multiple ClearSpeed chips from very high-level languages (VHLLs)
- Programming interface
- Performance
- Configurations

Star-P Basics:

Bridges the gap between desktop tools and parallel computing systems

Value proposition

- Rapid, interactive apps development
- Potent high-level parallel abstractions
- Minimize code changes
- High speed and/or large memory
- MATLAB® and Python clients today, R soon
- Scales to 100s of cores, >4TB memory
- Extensible with existing serial or MPI-parallel libraries





Using Star-P with MATLAB

Task-Parallel

- Iterations clearly separable
- Use Star-P's `ppeval`

```
%Generate the Fourier Transform on 10 degree spacing
angles = linspace(0,360,37);
%Serial Version
load('brain.mat','A');
for i = 1:length(angles);
    FFTangles(:,:,i) = genFFTangles(angles(i),A);
end
```

Data Parallel

- Large monolithic data
- Use Star-P's `*p` construct
 - Distributed attribute propagates to result variables

```
% explicitly parallel with *p
n=10000*p

% implicitly parallel
A = rand(n, n);

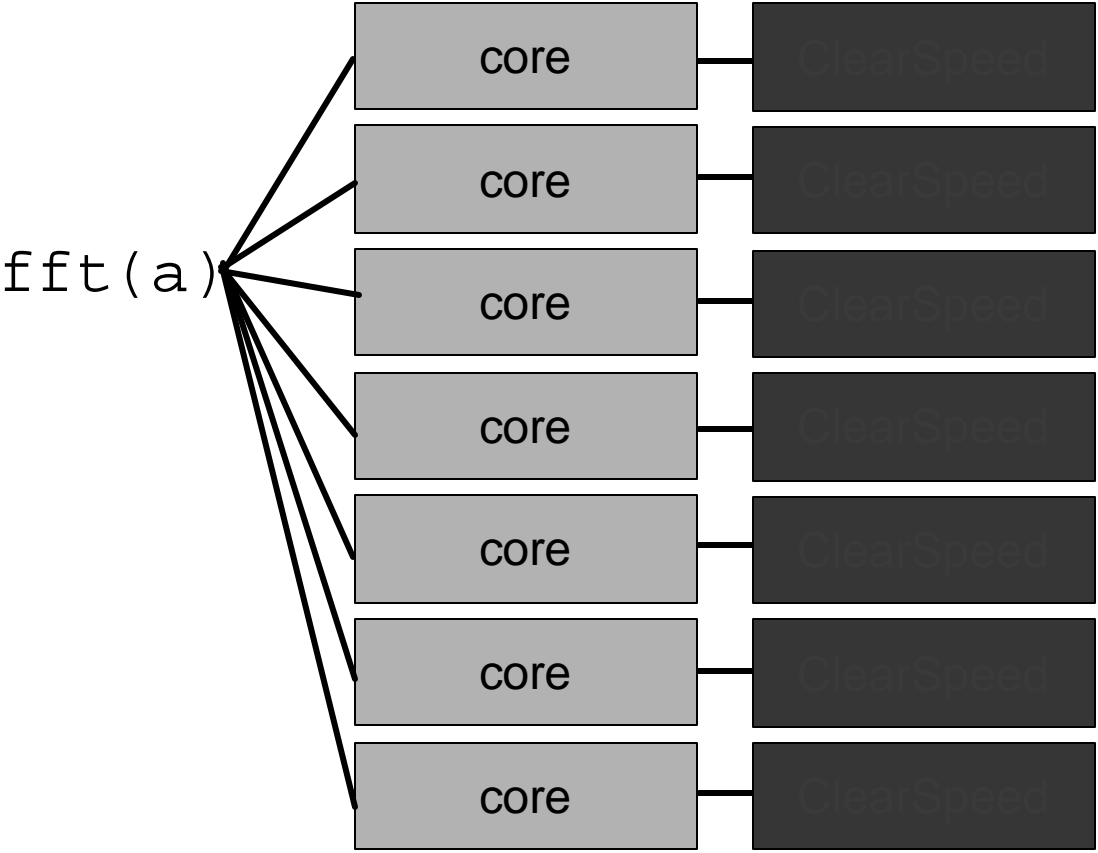
% implicitly parallel
x = randn(n, 1);

% implicitly parallel
y = zeros(size(x));

while norm(x-y) / norm(x) > 1e-11
    y = x;
    x = A*x;
    x = x / norm(x);
end;
```



Conceptual Motivation

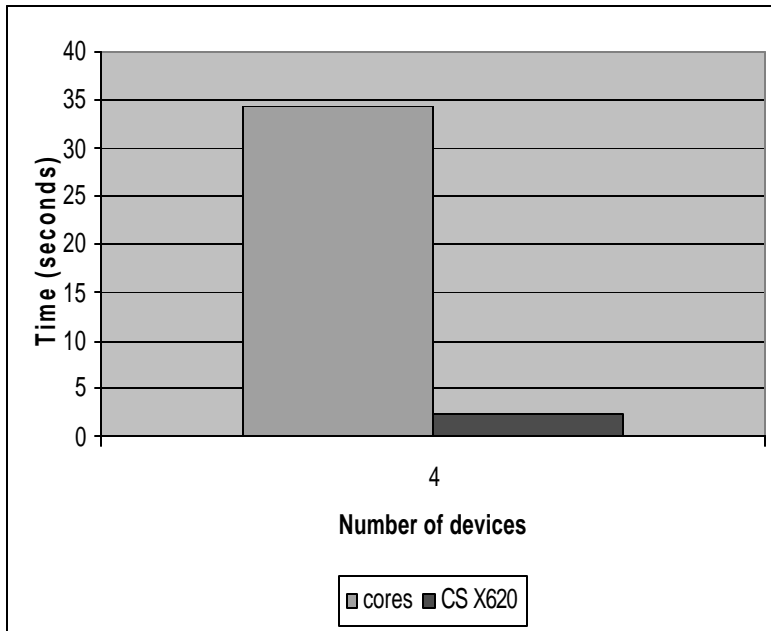


- Parallel programming is already hard enough
- Don't add an extra burden of low-level core? accelerator linkage
- Goal: ~0 changes to the VHLL program

Programming Interface

- For Star-P end-users
 - In shell rc file (*i.e.*, `.bashrc`) on the server system, ‘source’ a file that adds the ClearSpeed library to the search path. Then Star-P uses the ClearSpeed library automatically for subsequent sessions.
 - Zero VHLL source code changes to use accelerators
- For Star-P internal developers
 - ISC had to add the ability for a user to specify from where libraries (*e.g.*, BLAS) are to be loaded; should be sufficient for future ClearSpeed library additions
 - Otherwise, ClearSpeed interface just worked
 - ClearSpeed library makes threshold decisions; *i.e.*, small matrices don’t benefit

Performance: Matrix multiplication

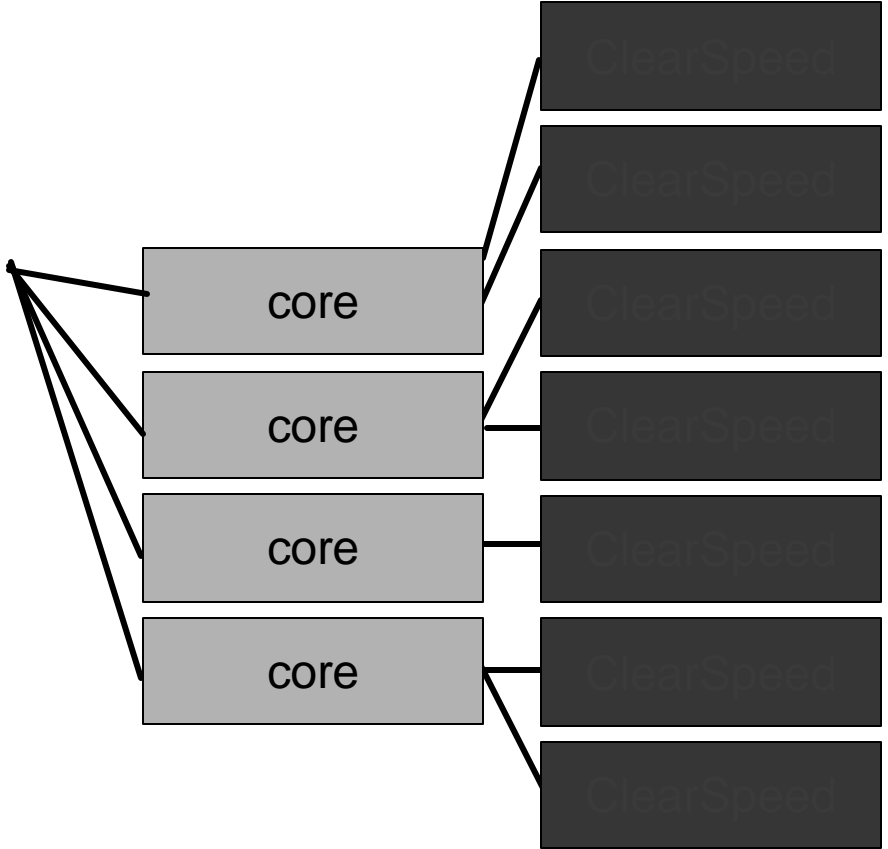
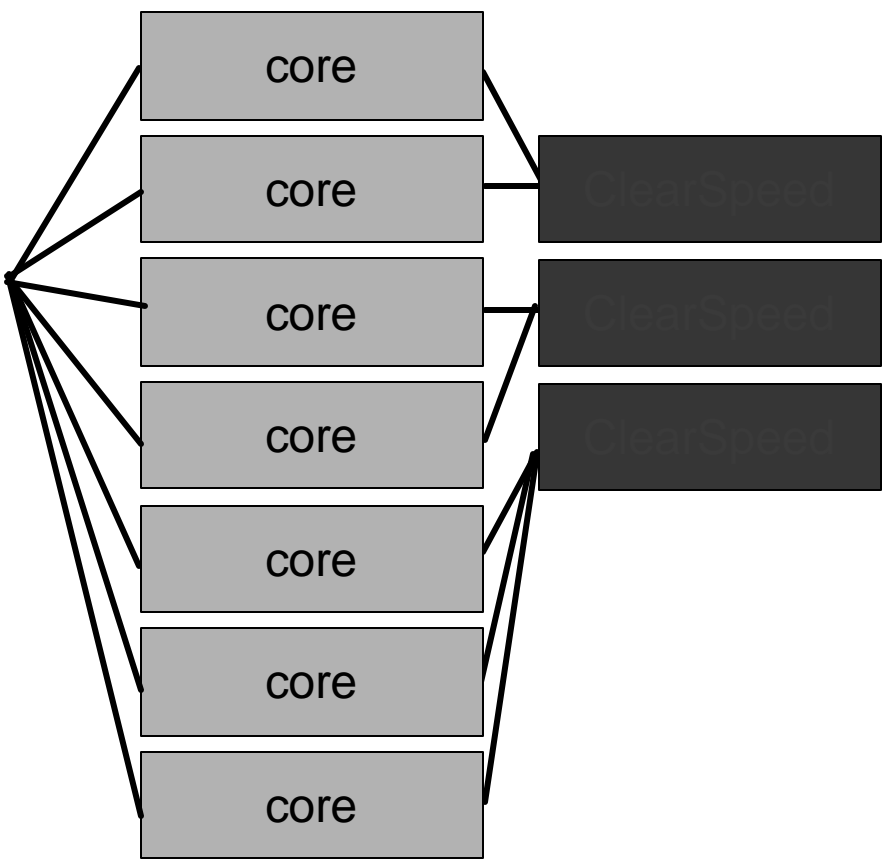


- Task-parallel, on each plane of the input array
- Using 4 cores and 4 ClearSpeed X620 boards
- 14.5X faster using X620s
- Code

```
A=rand(4096,4096,4*p);
tic, B=ppeval('*',A,A); toc
```
- Extends readily to other ClearSpeed library routines (BLAS, LAPACK, and FFTs)
- Data-parallel operations that are not purely local (e.g. matmul) will require communication between calls to ClearSpeed routines



Other Possible Configurations



Memory capacity and bandwidths make core/ClearSpeed balance application-dependent

What configuration do you need for your applications?

Summary

- Parallel acceleration demonstrated with excellent speed-up
- Only set-up changes to Star-P, no user code changes
- ClearSpeed/ISC software will get smarter about when to use accelerators
- Near-term, expected users are early adopters



For more info, see
www.InteractiveSupercomputing.com

INTER*CTIVE
supercomputing