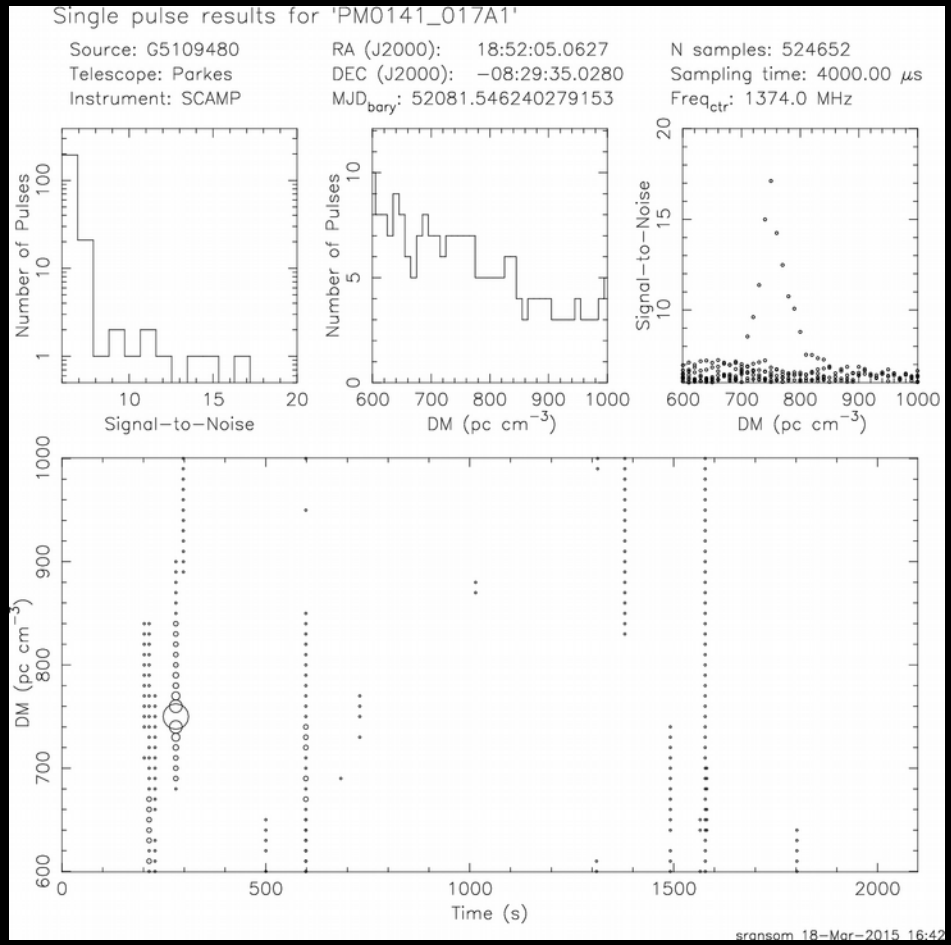
Current and Future Work on Fast Transient Search Algorithms

Scott Ransom

National Radio Astronomy Observatory / University of Virginia



Basic FRB Search Algorithm

- The following is done *per beam*
  - RFI characterization / excision
  - De-dispersion
  - Matched Filter Search (usually boxcar)
  - Candidate sifting
- Since there is no (or very little) integration, algorithm can be pipelined and realtime
- De-dispersion is by far the most expensive step

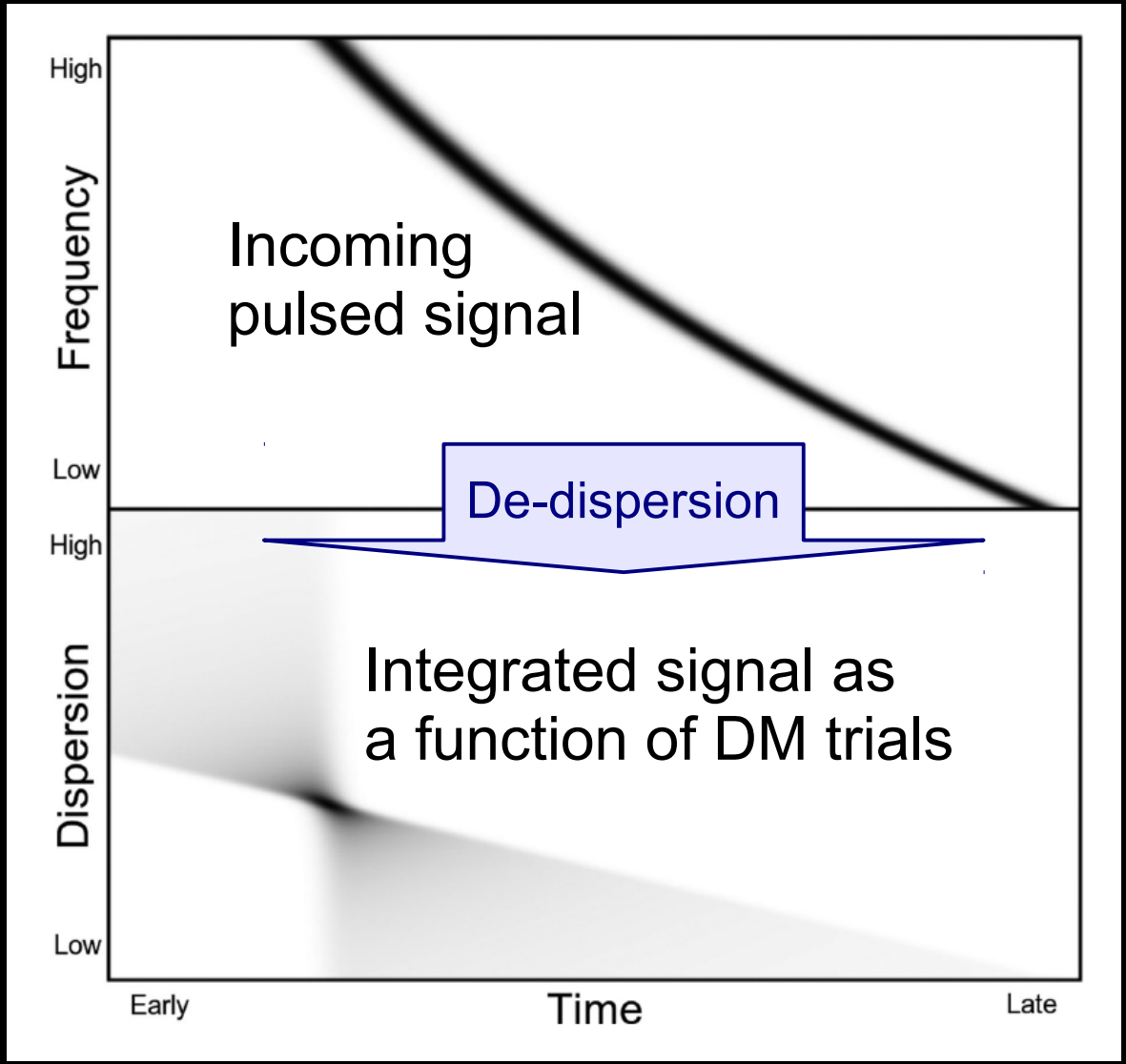
Dispersion

Lower frequency radio waves are delayed with respect to higher frequency radio waves by the ionized interstellar medium (ISM)

$\Delta t \propto DM\nu^{-2}$

(DM = Dispersion Measure)

- Need ~10^4 frequency channels
- DM for undiscovered pulsar/FRB is unknown
- Must search over ~few x 10^4 trial DMs!
- This multiplies data rate by factor of few
- De-dispersion is very I/O intensive



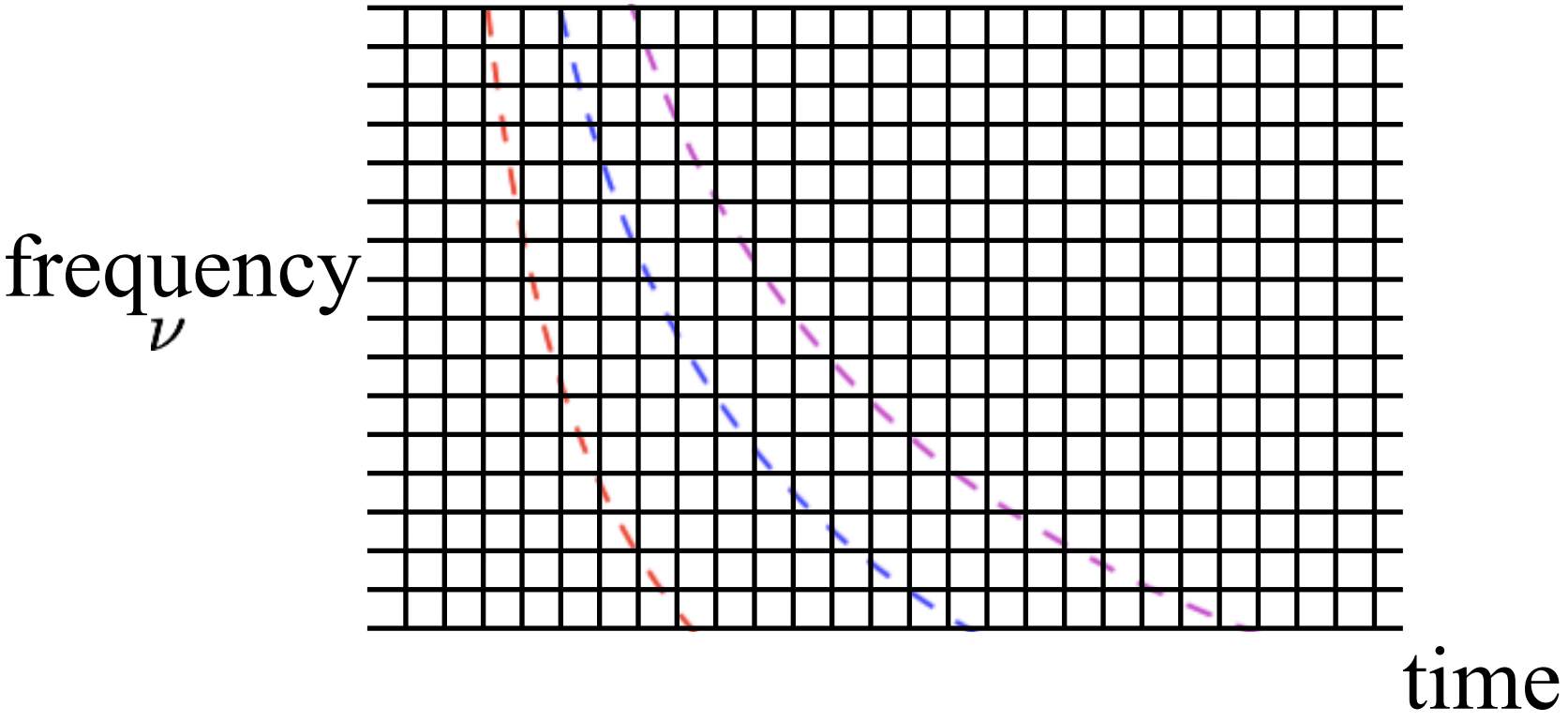
Barsdell et al 2012

De-dispersion State-of-the-Art

- All de-dispersion is limited by memory bandwith
  - Arithmetic intensity per memory access is low
- Brute force dedisp ~$O(N_{chan}^2)$
  - Often use GPUs:
    - *dedisp* (in HEIMDALL), ARTEMIS, Maggro code…
  - Very flexible (e.g. DM spacing, optimality)
- Tree Algorithms ~$O(N_{chan} \log_2 N_{chan})$
  - Known since early 1970s, but often ignored
  - New variants fix lots of “problems”
    - FDMT (Zackay et al. 2016), *bonsai* (Smith et al., in prep)

Incoherent dedispersion

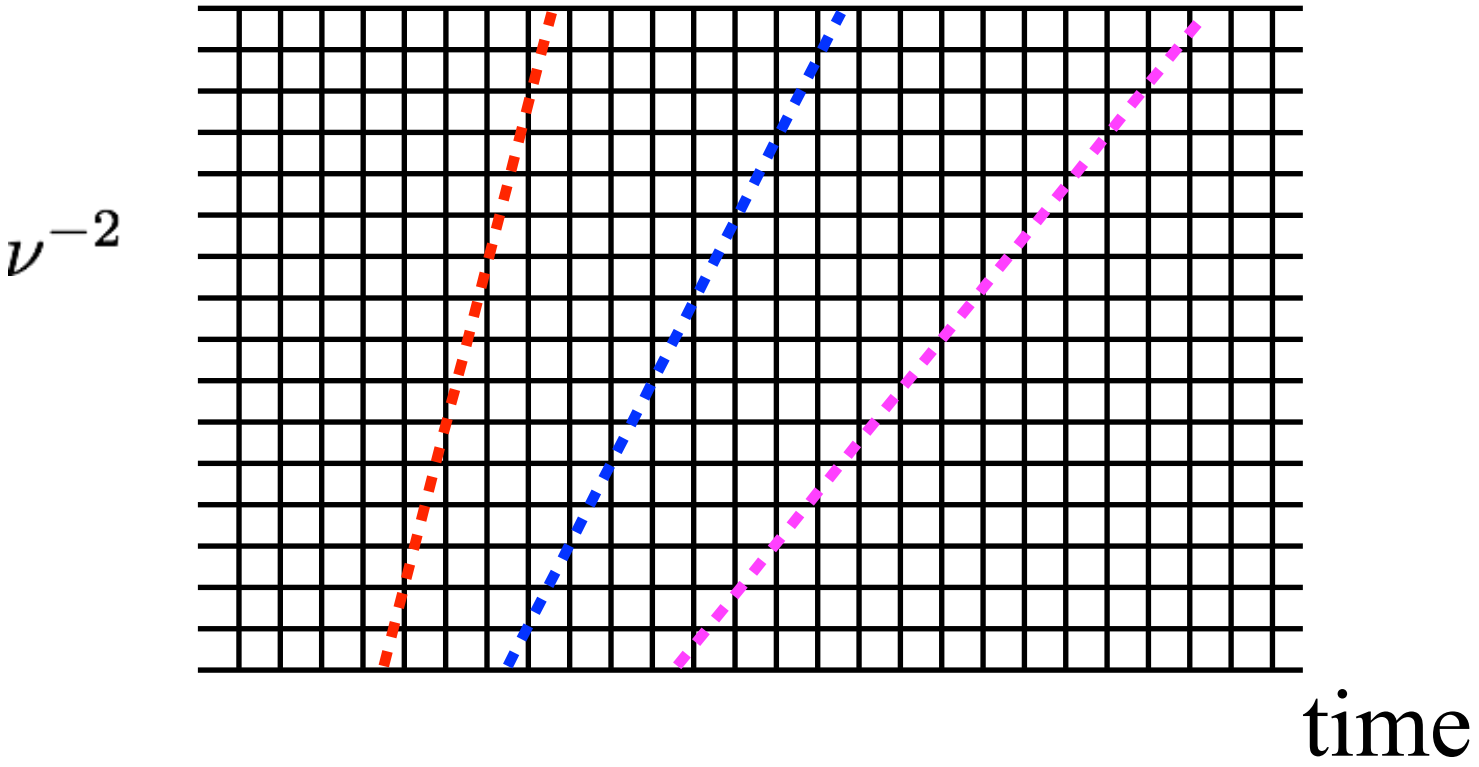
FRB backend incrementally receives a 2D array with (time, frequency) axes. We want to sum over all “tracks” with the shape shown.



Slide from Kendrick Smith

Incoherent dedispersion

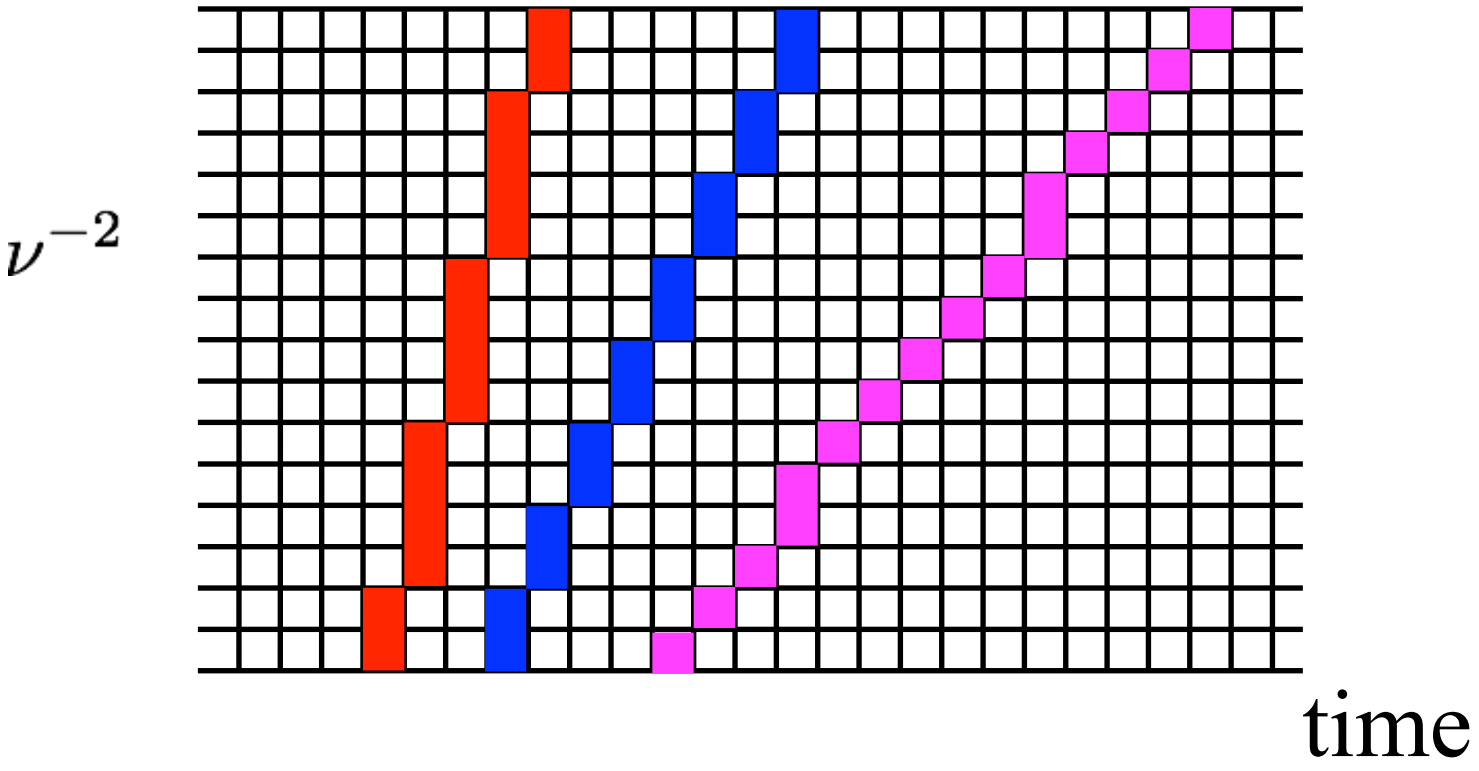
The first step is to change variables $\nu \rightarrow \nu^{-2}$
in order to transform the curves to straight lines.



Slide from Kendrick Smith

Tree dedispersion

The algorithm I'll describe is a "tree" algorithm which ends up approximating each straight-line track by a jagged sum of samples.



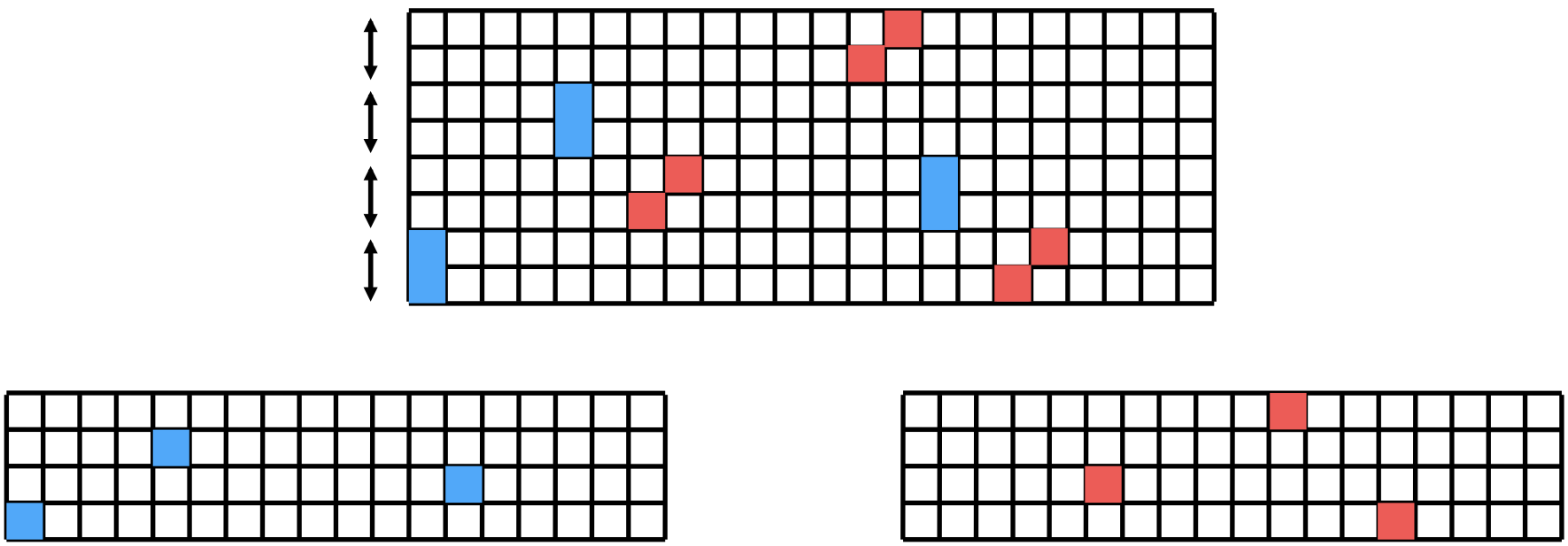
The sums are built up recursively.

Slide from Kendrick Smith

Tree dedispersion

First iteration: group channels in pairs. Within each pair, we form all "vertical" sums (blue) and "diagonal" sums (red).

Output is two arrays, each half the size of the input array.

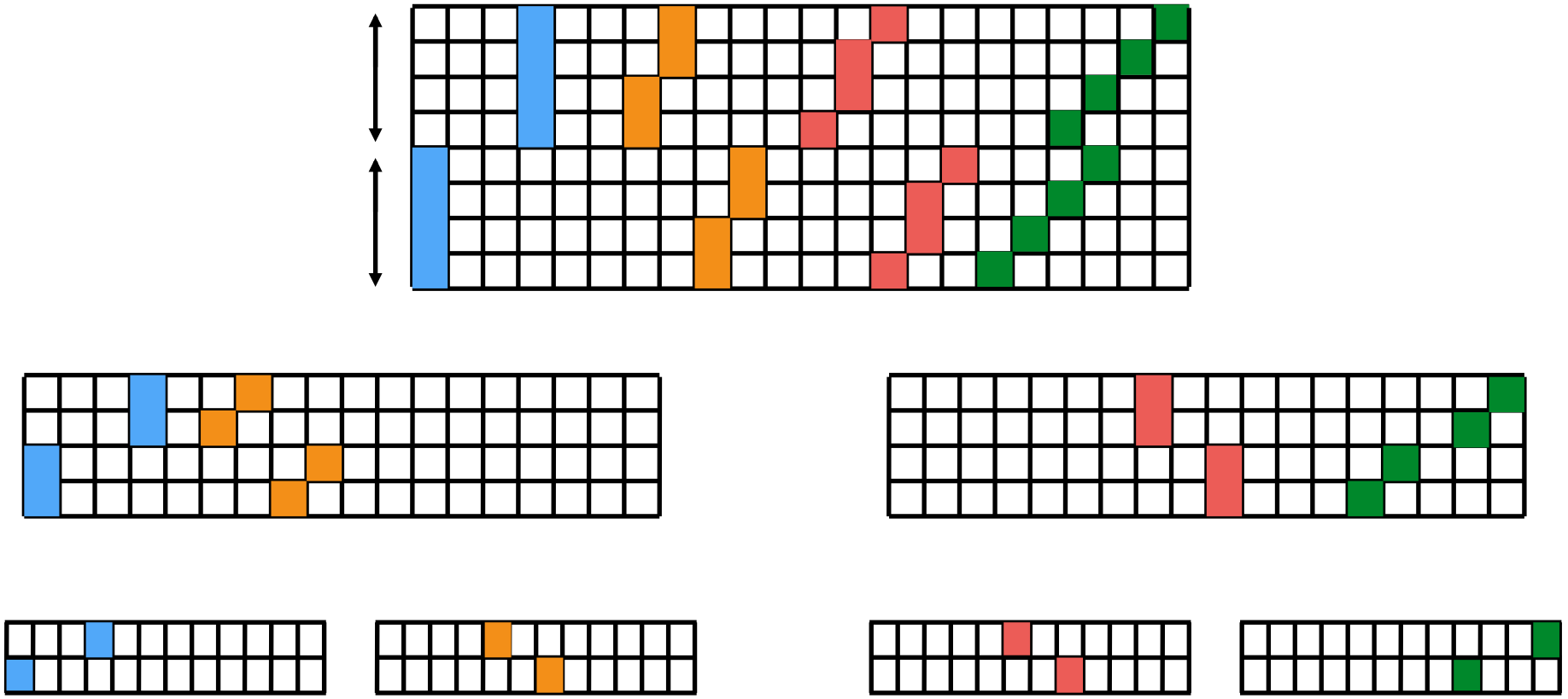


Slide from Kendrick Smith

Tree dedispersion

Second iteration: sum pairs into “pairs of pairs”.

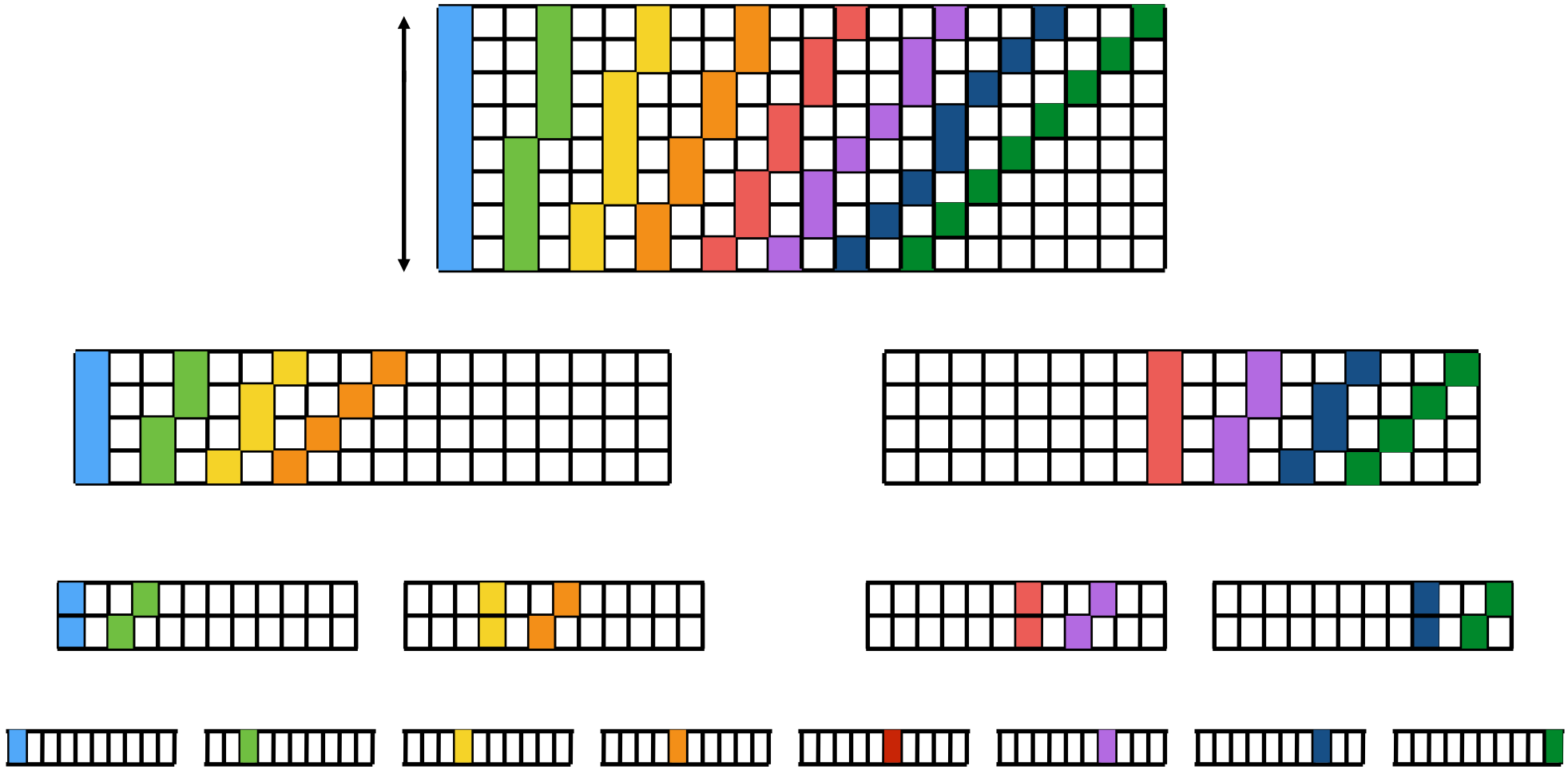
Frequency channels have now been merged in quadruples.
Within each quadruple, there are four possible sums.



Slide from Kendrick Smith

Tree dedispersion

Last iteration: all channels summed.



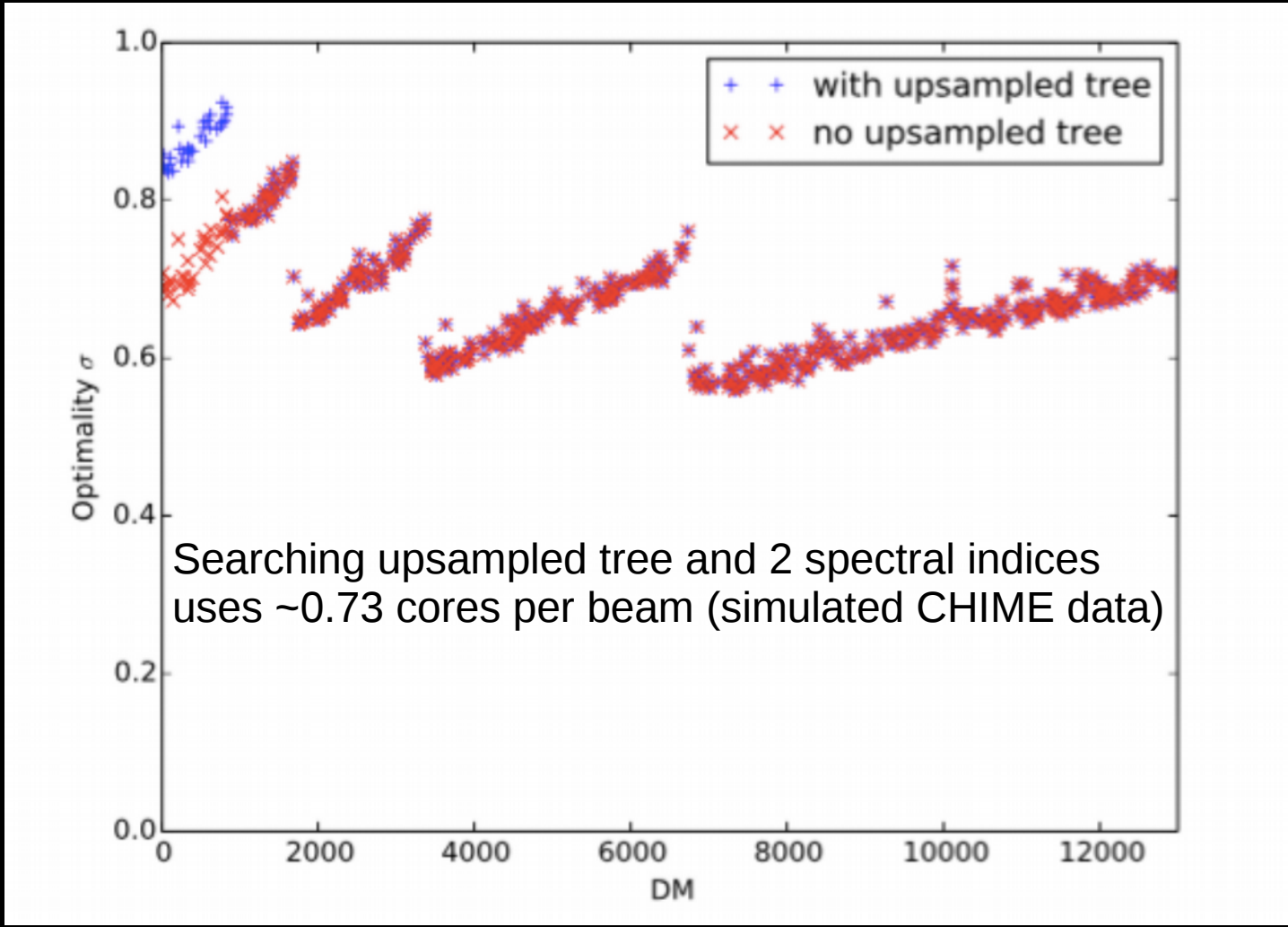
Slide from Kendrick Smith

bonsai by Kendrick Smith

- Highly optimized, multi-core, CPU-based, tree code (based on already good code by Jonathan Sievers)
- Mathematically proved that a “proper” tree-code could be made arbitrarily close to optimum
  - Increase N_{chan}, “smear” channels, upsampled (in time) tree
- Showed it could be generalized to search over spectral indices, scattering index, intrinsic width
- “Blocked” formulation allow extreme cache friendliness, better parallelism, and partial-band triggers
- Hand-optimized: vector intrinsics, linear speedup on multicore, special techniques for memory BW

Extensive Monte-Carlo Testing

- Random pulse injections with various pulse parameters (phase, width, scattering, spectral index)



Dedispersion Summary

- Because of low arithmetic intensity, GPUs are *not* the end-all-be-all solution for dedispersion
  - Rise of multi-core CPUs and GPU “extra” power-usage both contribute to this
- A better algorithm (i.e. “proper” tree-dedisp) makes a huge difference
- *bonsai* will be released on github soon

Dealing with Many Beams at Once

- Each beam is independent
  - Leads to pipelined processing per beam
  - Requires zero communications between beams
  - “pleasingly” parallel
- RFI is huge problem, and contributes to some or all beams (i.e. inter-beam correlations)
- Inter-beam communication to deal with RFI can cause significant loss of parallelism
  - Need to ensure communication costs are low
  - This is typically handled in the “sifting” stage

FRB Searching with Interferometers

- Generate sky pixels vs time by 1 of 2 ways:
  - Beamforming: sum delayed antenna voltages via hardware implementation or highly-vectorized DSP code
    - De-disperse the pixel channels via tree-dedispersion
  - Imaging: transform post-correlator visibilities into an image via FFT
    - De-disperse the visibilities via tree-dedispersion
- Casey Law's *rtpipe* for *realfast* is a great example
- Potentially innovative ways to use autocorrelations (or other correlator products) for RFI removal

Summary

- For single beams, de-dispersion is most costly step
- For interferometers, can be dedisp or imaging
- Use NlogN algorithms whenever possible!
  - Tree-dispersion has seen many improvements
- GPUs are no longer the “obvious” answer
  - Need to carefully check power requirements
  - Weigh pros/cons of CPU speed vs. flexibility and GPU speed vs. programming difficulties and power/cost
- Auto-tune your algorithm to your hardware! (e.g. very cool work by Alessio Sclocco et al. 2016).
- While not trivial, FRB searching is still *much* simpler than searching for binary millisecond pulsars….