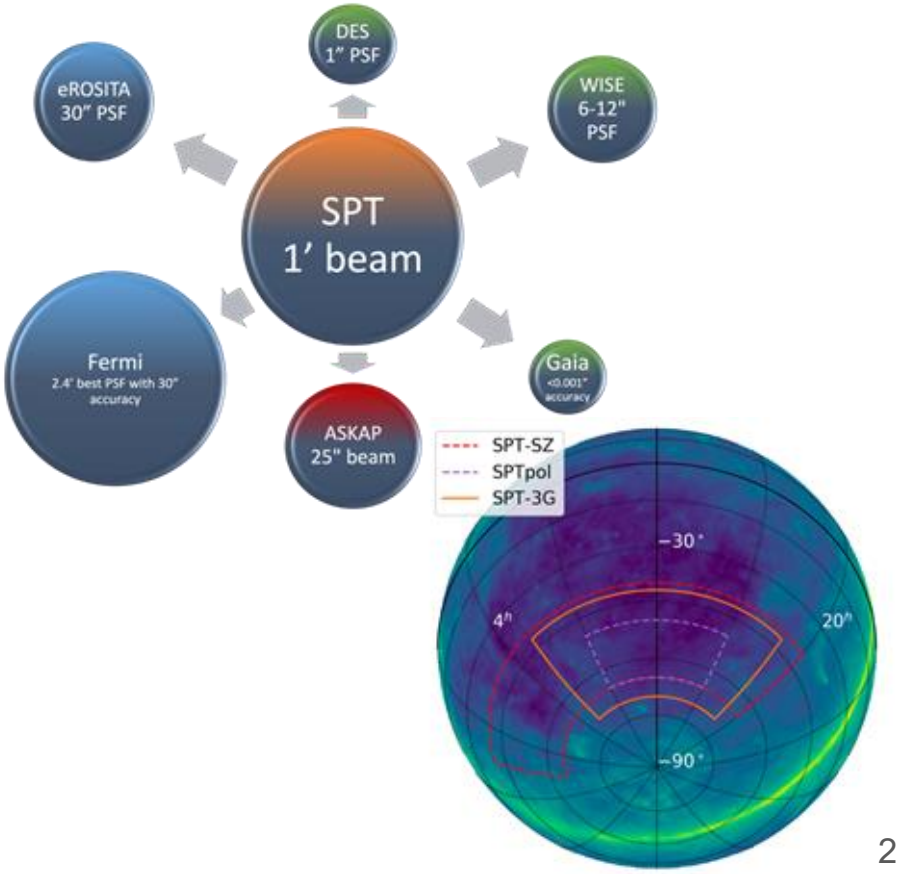
# Automating Bayesian Inference of Millimeter Source Association

PIs: Joaquin Vieira and Billy Moses

Kedar Phadke, Vimarsh Sathia, Brant Qian

# Millimeter Point Source Association

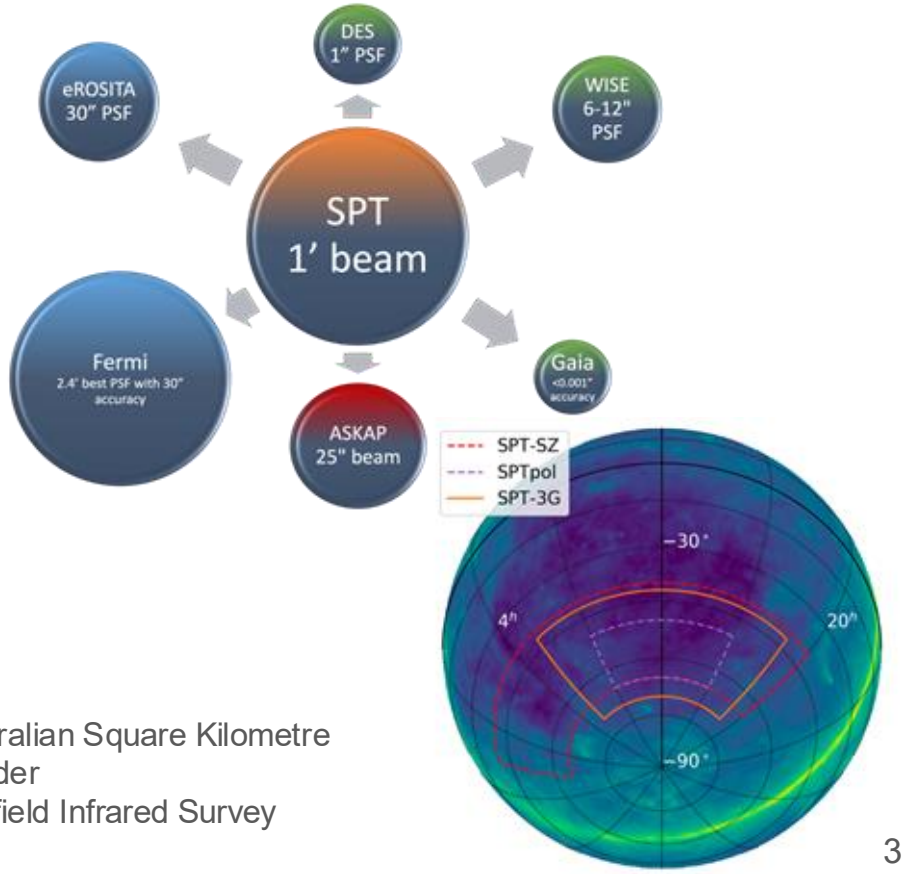
- Characterize sources detected at millimeter wavelengths
- Develop a forward model for the emission
- Fast Bayesian inference including prior from forward model to associate sources at other wavelengths.




Funded by the U.S. National Science Foundation and the Simons Foundation

# Millimeter Point Source Association

- ~4500 sources in the first South Pole Telescope catalog (SPT-SZ)
- ~30000 sources in the new SPT-3G (3rd generation) catalog
- ~265000 sources in the same area in ASKAP [radio] catalog
- ~100000 in WISE band 4




Funded by the U.S. National Science Foundation and the Simons Foundation

ASKAP: Australian Square Kilometre Array Pathfinder
WISE: Wide-field Infrared Survey Explorer

# The South Pole Telescope

- 10 m off-axis mm-wave telescope
- Observes at 1.4, 2, 3 mm + polarization
- Main science goal to observe the cosmic microwave background
- ~1 arcminute beam (1.22*$\lambda$/D)

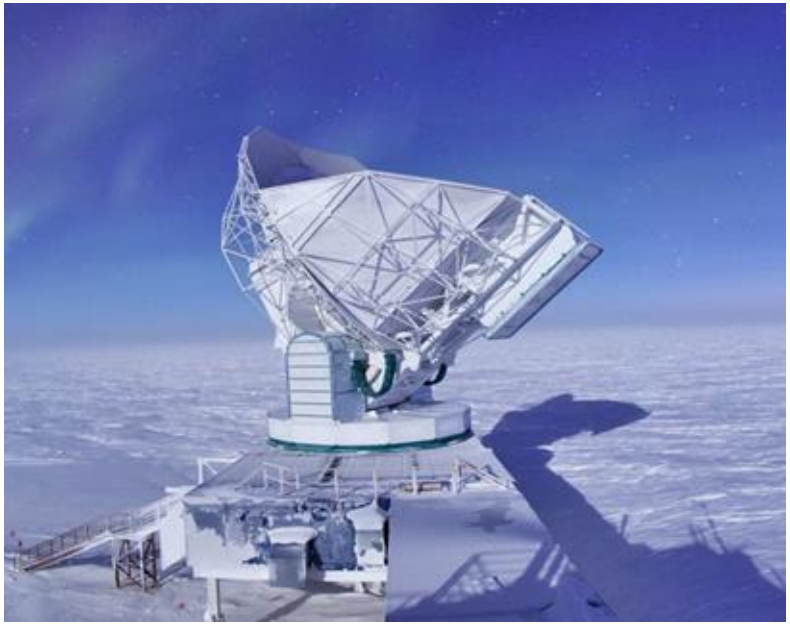
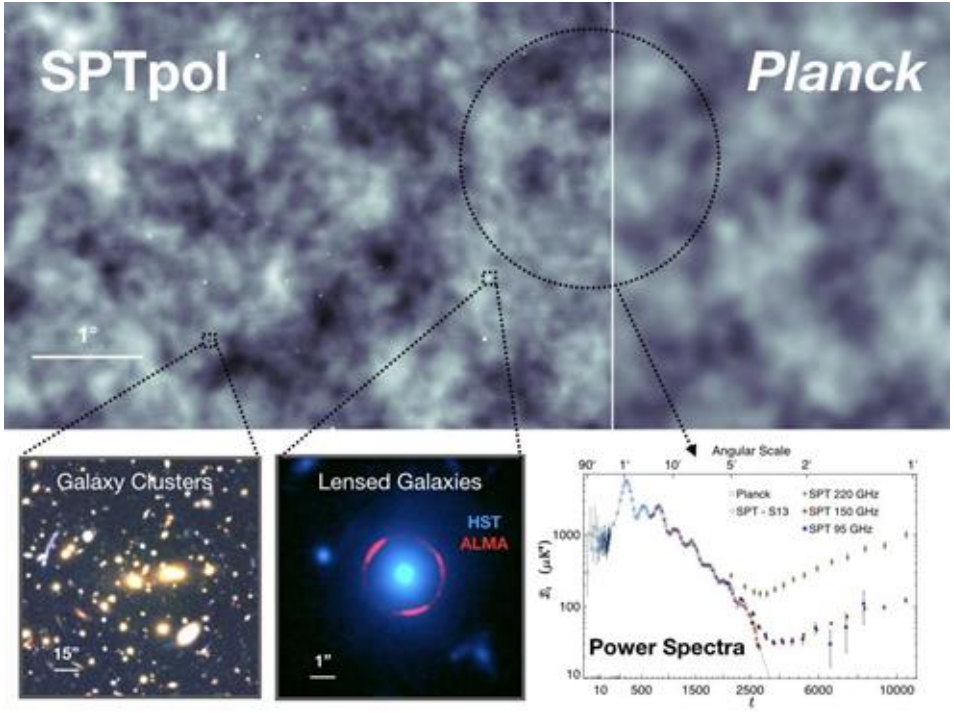


Photo credit: Jason Gallichio

Funded by the U.S. National Science Foundation and the Simons Foundation

# The South Pole Telescope

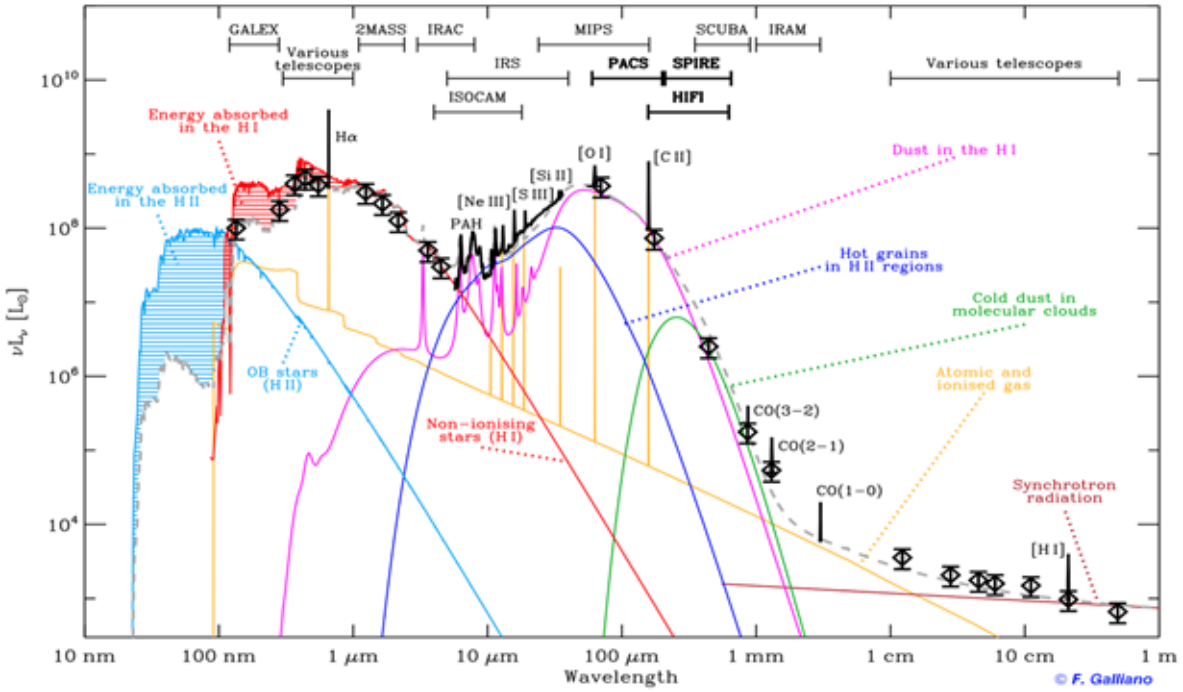
- Dark spots -> Galaxy clusters
- Emissive sources -> Galaxies/AGN
- Our focus in this project is to characterize the emissive sources




https://astro.fnal.gov/science/cmbr/spt-3g/

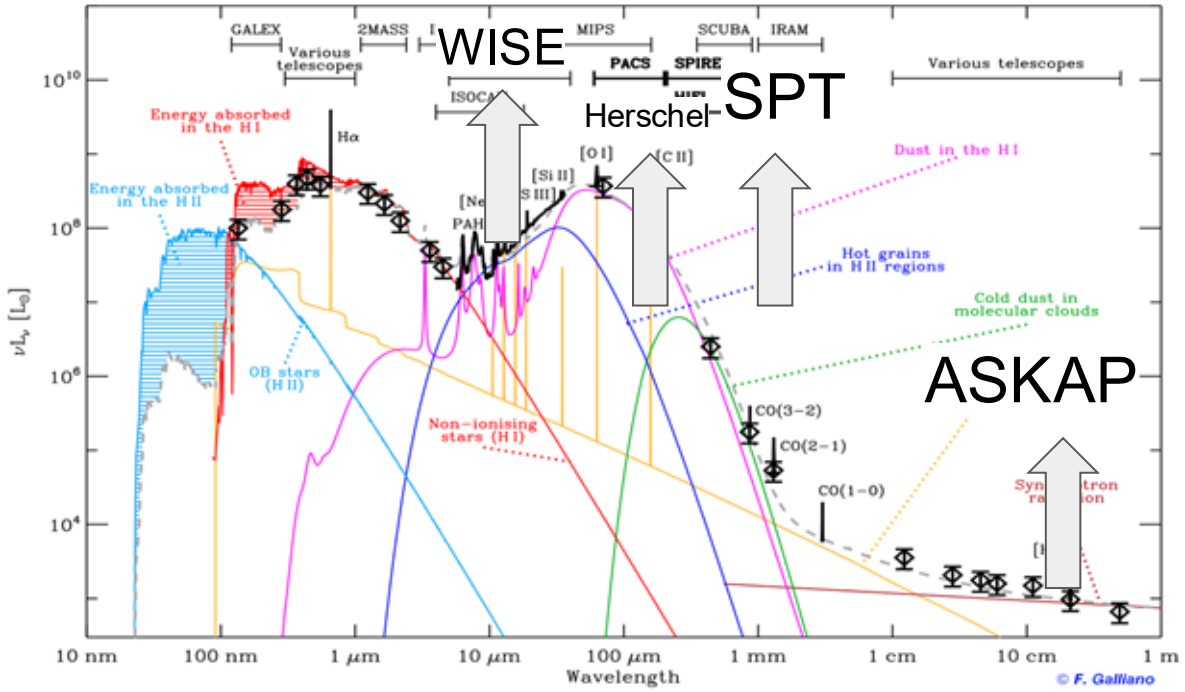
Funded by the U.S. National Science Foundation and the Simons Foundation

# Spectral Energy Distribution of a galaxy



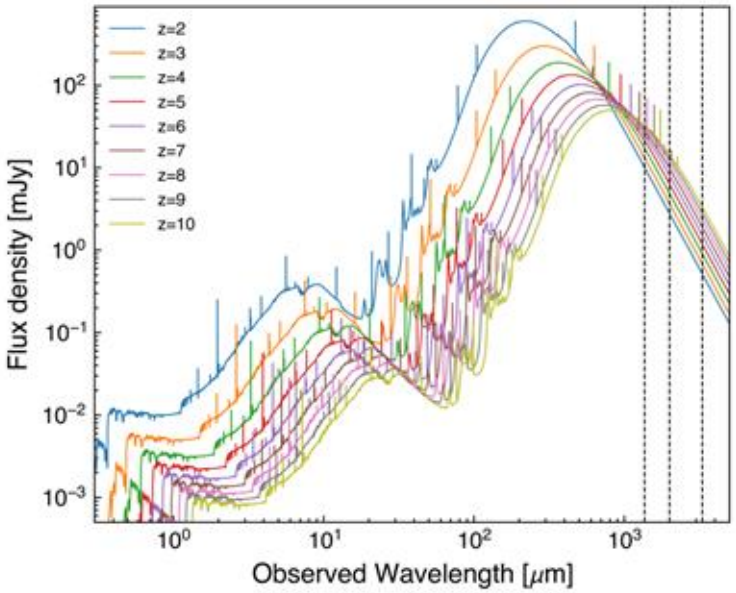

Funded by the U.S. National Science Foundation and the Simons Foundation

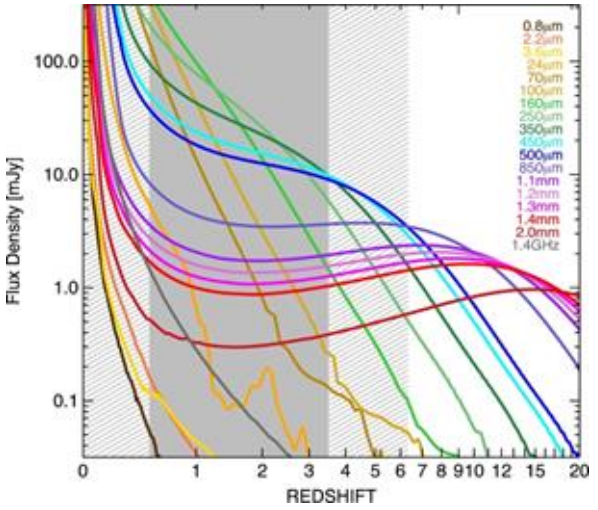
# Spectral Energy Distribution of a galaxy




Funded by the U.S. National Science Foundation and the Simons Foundation

# Negative k-correction



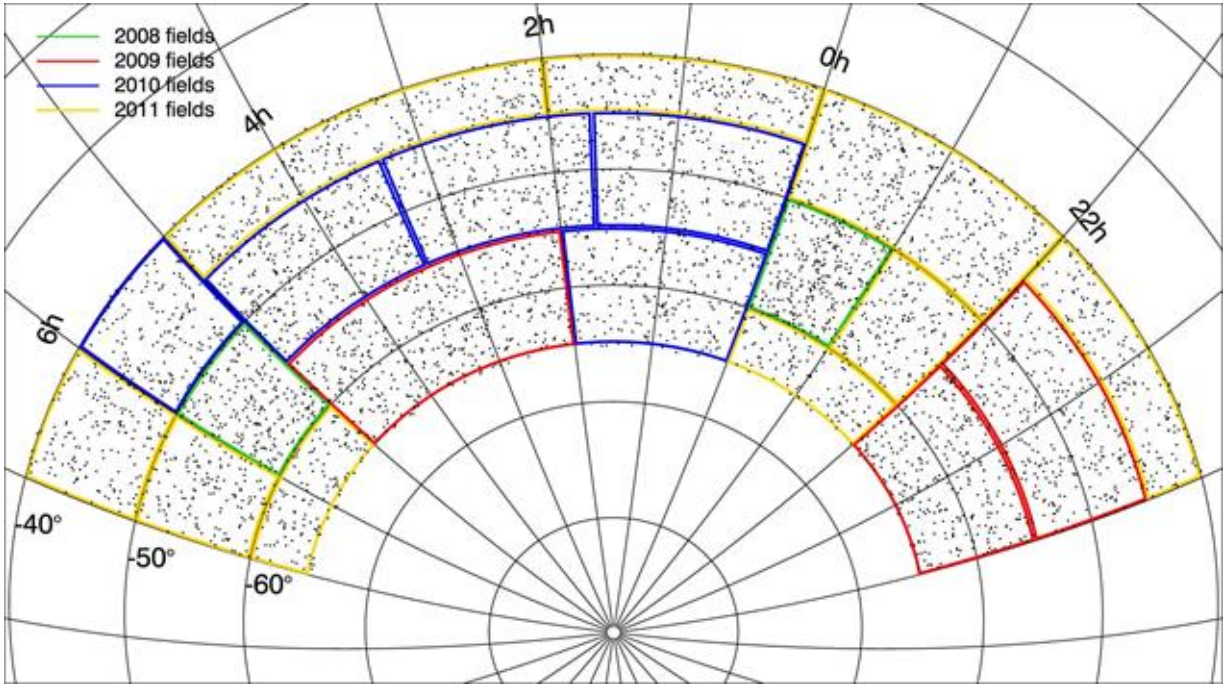

Casey et al. 2014 Figure 3

Take a star forming galaxy/ mm source and keep moving farther away.

Funded by the U.S. National Science Foundation and the Simons Foundation

# The SPT-SZ catalog

- 4845 sources in the first South Pole Telescope catalog (SPT-SZ)
- 3980 synchrotron dominated sources (80%)
- 865 dust dominated sources (20%)

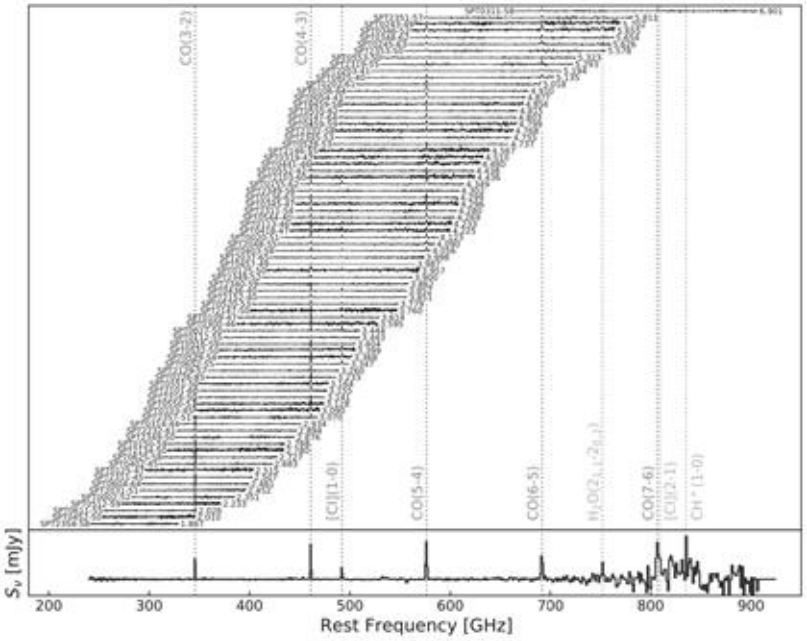



Everett et al. 2020

Funded by the U.S. National Science Foundation and the Simons Foundation

# SPT selected Dusty Star Forming Galaxies

- 81 high-redshift dusty star forming galaxies
- 1.8<z<6.9
- Pushing for higher redshift with SPT-3G to constrain dust production mechanisms
- Quantify contribution of dusty star formation to the total star formation history of the universe

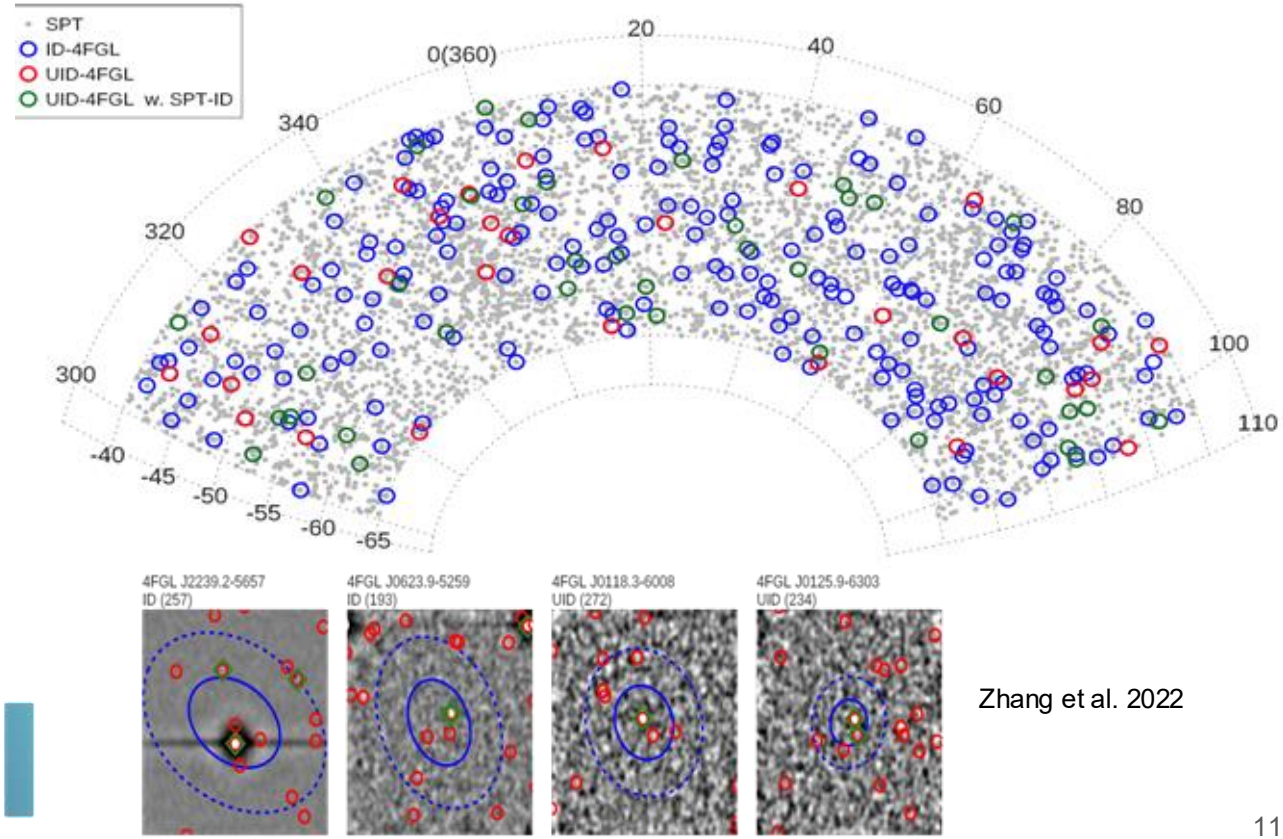



Reuter et al. 2020

Funded by the U.S. National Science Foundation and the Simons Foundation

# SPT selected Active Galactic Nuclei (AGN)

- Identified counterparts for 40 previously unassociated Fermi γ-ray sources using frequentist approach
- Demonstrated flux correlation between millimeter and γ-ray emission

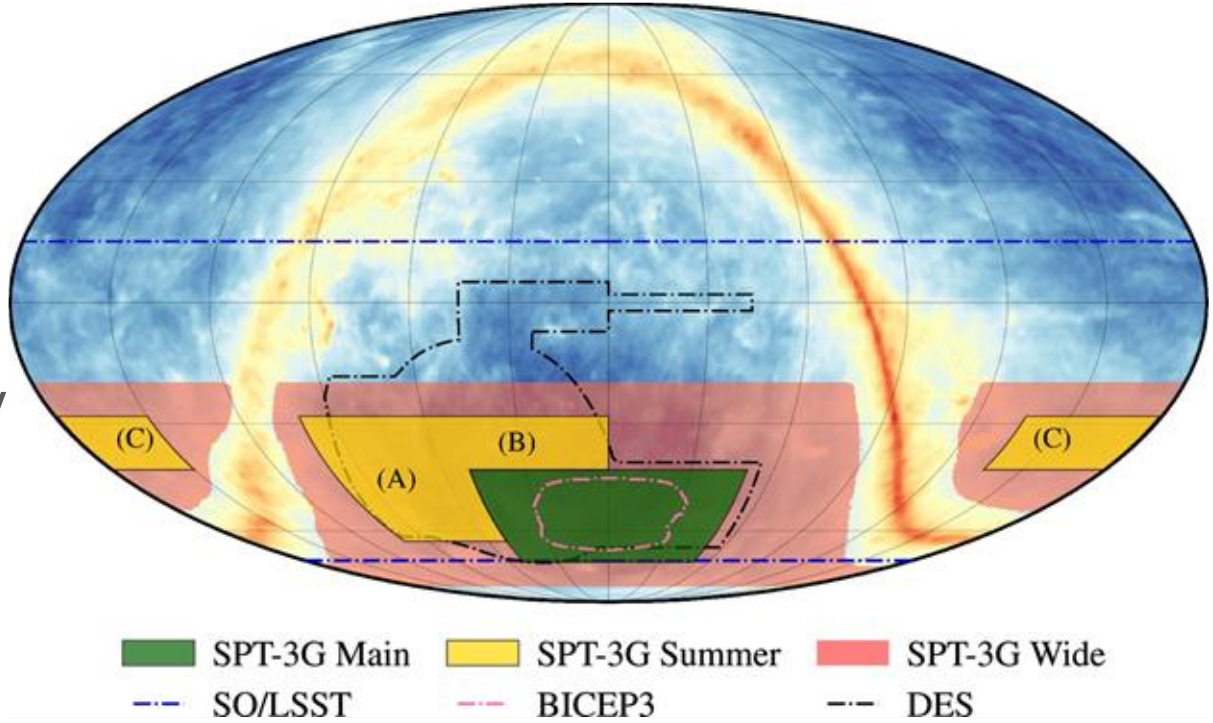



Zhang et al. 2022

Funded by the U.S. National Science Foundation and the Simons Foundation

# Current SPT status

- 1500 sq degrees of SPT-3G Main
- 10k sq degrees of SPT-3G Main+ Wide
- Overlaps with many southern hemisphere surveys

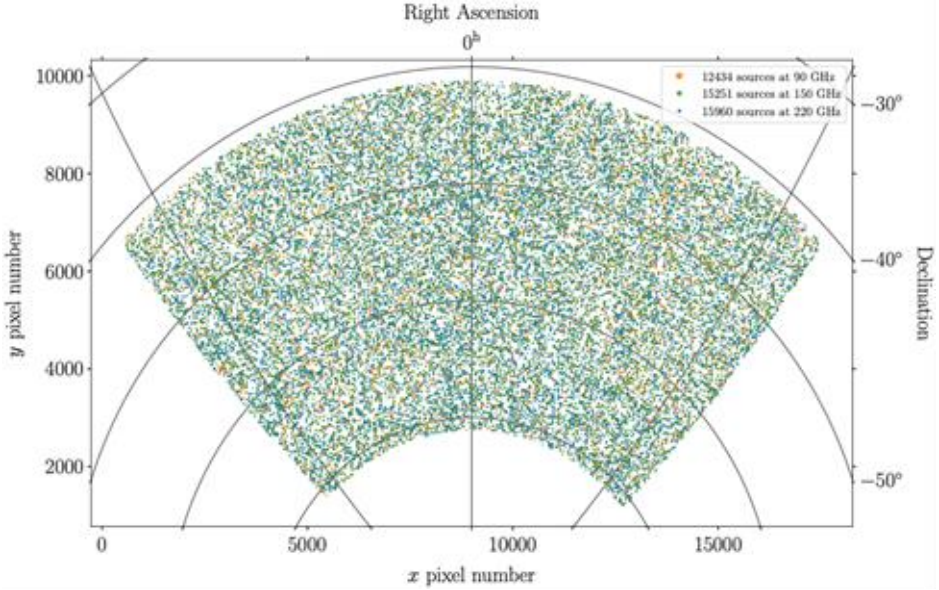



Prabhu et al. 2024

Funded by the U.S. National Science Foundation and the Simons Foundation

# The SPT-3G Main source catalog

- Based on 5 year data 2019-2023
- ~4x more sensitive at 1.4 mm
- Change in demographics of the sources
- Nearly 50-50 split between synchrotron and dust dominated sources.
- Need new techniques to select high-redshift dusty star forming galaxies.




Archipley et al. in prep. 2026

Funded by the U.S. National Science Foundation and the Simons Foundation

# Bayesian inference for source association using NWAY

N: Number of sources.
k: Different catalogs.
P(H): Prior probability of an association.
ν: Source density.
Ω: Sky area.
P(H|D): Posterior probability.
$\sigma$: Position uncertainty.
p_any: Probability that source has any counterpart.

$$P(H) = N_1 / \prod_{i=1}^{k} N_i = 1 / \prod_{i=2}^{k} N_i = 1 / \prod_{i=2}^{k} \nu_i \Omega_i \,. \qquad \text{eq. (1)}$$

$$P(H|D) \propto P(H) \times P(D|H) \,. \qquad \text{eq. (2)}$$

$$P(D|H) = 2^{k-1} \frac{\prod \sigma_i^{-2}}{\sum \sigma_i^{-2}} \exp\left\{ -\frac{\sum_{i<j} \phi_{ij} \sigma_j^{-2} \sigma_i^{-2}}{2 \sum \sigma_i^{-2}} \right\} \qquad \text{eq. (3)}$$

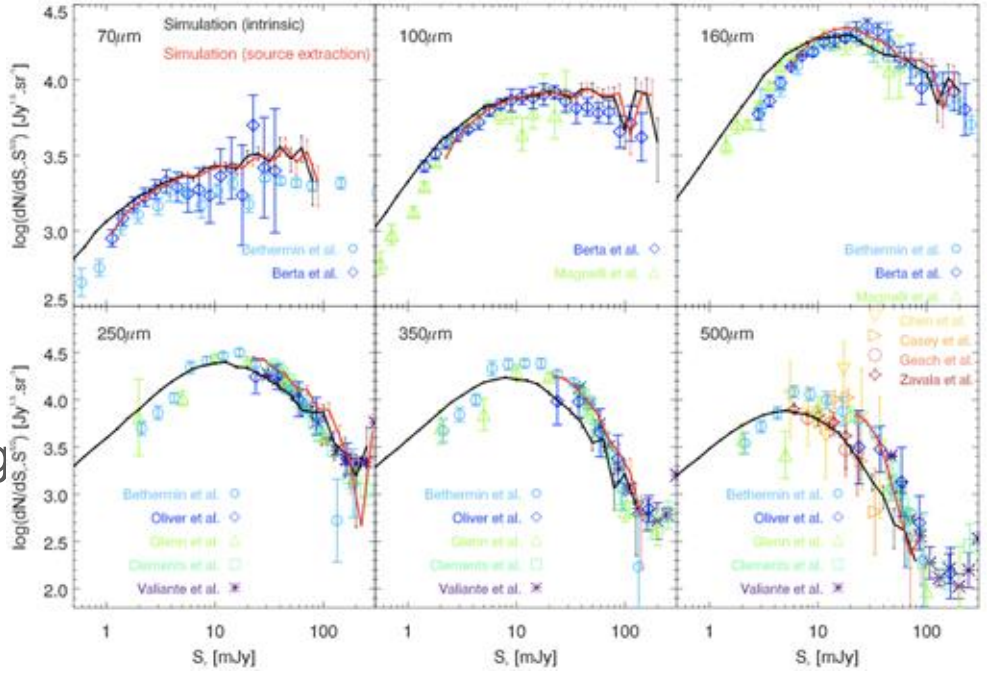
$$p\_any = 1 - P(H_0|D) / \sum_i P(H_i|D) \qquad \text{eq. (4)}$$

Funded by the U.S. National Science Foundation and the Simons Foundation

Budavári T.and Szalay A. S., 2008
Salvato, M., Buchner, J., Budavári, T., et al. 2018
Salvato, M., Wolf, J., Dwelly, T., et al. 2022

# Forward model of the source catalog using SIDES

- Simulated Infrared Dusty Extragalactic Sky (SIDES)
- Built using Bolshoi-Planck dark-matter simulation
- 2 deg² simulation from IR to submillimeter
- Populated via abundance matching
- Reproduces observed number counts




Bethermin et al.. 2017, 2022

Funded by the U.S. National Science Foundation and the Simons Foundation

# Goal of this project

### Forward model

Generate source catalog at multiple wavelengths. Build on the existing simulation (SIDES) and add new features.

### Bayesian inference for source association

Probability of a source to be associated with sources in catalogs across the electromagnetic spectrum

### Differentiable compiler

Using Reactant to optimize the forward modeling and bayesian inference we get fast outputs

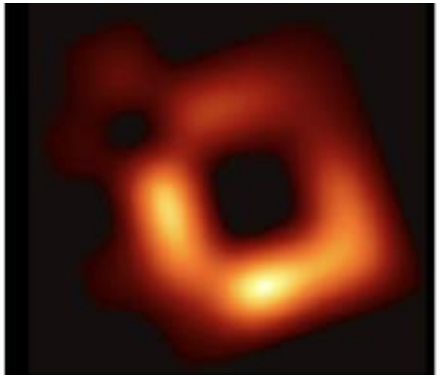
### Automating Bayesian inference for (mm) source association

The flow chart should work for other catalogs provided appropriate prior is generated using a forward model.

Funded by the U.S. National Science Foundation and the Simons Foundation

# Bayesian Inference is great….but expensive!

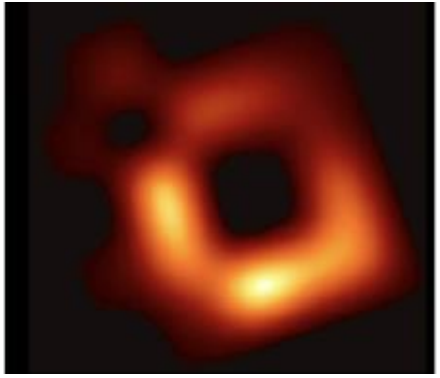
- **Bayesian inference** infers unknown from data
- Useful as it combines data + prior knowledge, even when data are limited or noisy!
- However, parameter space can be **large**, and inference needs **lots of** forward model runs (often with gradients)!



Funded by the U.S. National Science Foundation and the Simons Foundation

# Bayesian Inference is great….but expensive!

- **Bayesian inference** infers unknown from data
- Useful as it combines data + prior knowledge, even when data are limited or noisy!
- However, parameter space can be **large**, and inference needs **lots of** forward model runs (often with gradients)!

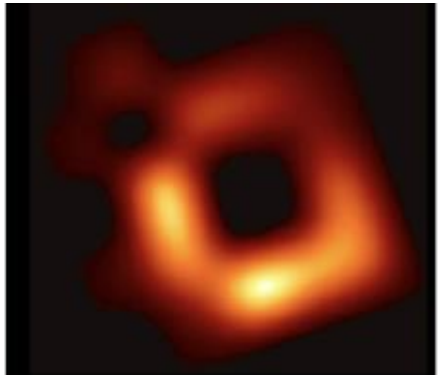


Reconstructed image of M87 ~1 week on cluster

SkAI
Funded by the U.S. National Science Foundation and the Simons Foundation

# Bayesian Inference is great….but expensive!

- **Bayesian inference** infers unknown from data
- Useful as it combines data + prior knowledge, even when data are limited or noisy!
- However, parameter space can be **large**, and inference needs **lots of** forward model runs (often with gradients)!



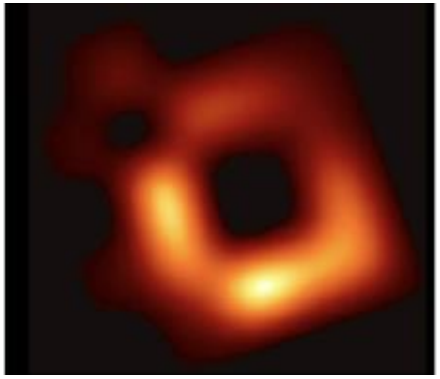
Reconstructed image of M87 ~1 week on cluster

```
result = None
best_loss = float('inf')
for _ in range(N):
    parameters = random()
    loss = grad(forward)(parameters)
    if loss < best_loss:
        best_loss = loss
        result = parameters
```

Funded by the U.S. National Science Foundation and the Simons Foundation

# Bayesian Inference is great….but expensive!

- **Bayesian inference** infers unknown from data
- Useful as it combines data + prior knowledge, even when data are limited or noisy!
- However, parameter space can be **large**, and inference needs **lots of** forward model runs (often with gradients)!



Reconstructed image of M87 ~1 week on cluster

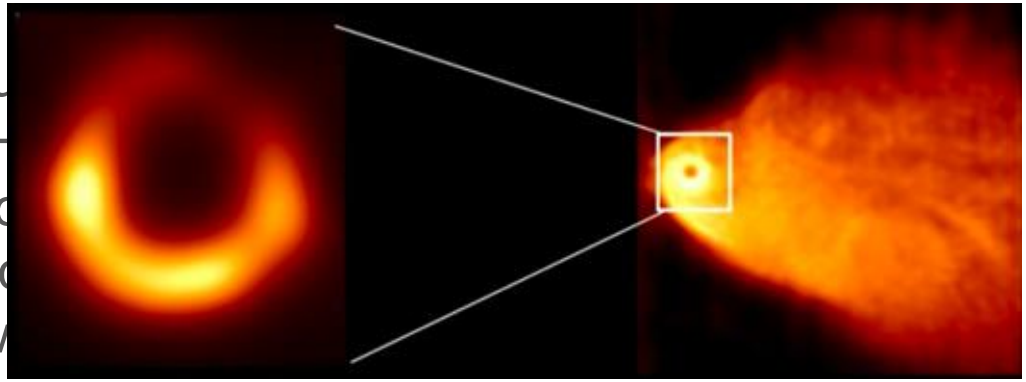
Expensive

```
result = None
best_loss = float('inf')
for _ in range(N):
    parameters = random()
    loss = grad(forward)(parameters)
    if loss < best_loss:
        best_loss = loss
        result = parameters
```

Funded by the U.S. National Science Foundation and the Simons Foundation

# Bayesian Inference is great….but expensive!

- **Bayesian inference** infers u
- Useful as it combines data +
  even when data are limited c
- However, parameter space c
  inference needs **lots of** forw
  (often with gradients)!



With Enzyme differentiation:
**1 hour on 1 thread**

Expensive

```
result = None
best_loss = float('inf')
for _ in range(N):
    parameters = random()
    loss = grad(forward)(parameters)
    if loss < best_loss:
        best_loss = loss
        result = parameters
```

Funded by the U.S. National Science Foundation and the Simons Foundation

# Code You Want to Write != Code You Want to Run

- Scientific (and particularly astronomy) code is often written to resemble the fundamental equations to ensure correctness.
- Direct execution of the mathematical theory may often result in unnecessary work and suboptimal mapping to the hardware which must execute it.
- Even when this is taken into account, the code must be constantly re-written for each type and generation of hardware (CPU, 3090 GPU, 4090 GPU, 5090 GPU, TPUv1, TPUv2, …)

SkAI
Funded by the U.S. National Science Foundation and the Simons Foundation

# Code You Want to Write != Code You Want to Run

- Missing Algorithmic Optimizations
  - Example: expensive op **factored out**!
- Linear Algebra Optimizations
  - `diag(X) * Y + Z → X ⊙ slice(Y) + Z`
- Hardware optimizations
  - Rewrite to kernel which runs on devices 1, 2, 3, …, N, …

```
float[100][100] X;
for(i=1:10_000) {
    float alpha = rand();
    loss = cholesky(alpha * X)
    if(loss < best) update();
}
```

SkAI
Funded by the U.S. National Science Foundation and the Simons Foundation

# Code You Want to Write != Code You Want to Run

- Missing Algorithmic Optimizations
  - Example: expensive op **factored out**!
- Linear Algebra Optimizations
  - `diag(X) * Y + Z → X ⊙ slice(Y) + Z`
- Hardware optimizations
  - Rewrite to kernel which runs on devices 1, 2, 3, …, N, …

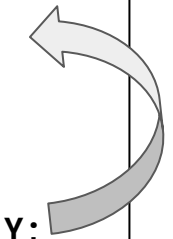
```
float[100][100] X;
for(i=1:10_000) {
    float alpha = rand();
    loss = sqrt(alpha) *
           cholesky(X)
    if(loss < best) update();
}
```

SkAI
Funded by the U.S. National Science Foundation and the Simons Foundation

# Code You Want to Write != Code You Want to Run

- Missing Algorithmic Optimizations
  - Example: expensive op **factored out**!
- Linear Algebra Optimizations
  - `diag(X) * Y + Z → X ⊙ slice(Y) + Z`
- Hardware optimizations
  - Rewrite to kernel which runs on devices 1, 2, 3, …, N, …

```
float[100][100] X;
Y = cholesky(X);
for(i=1:10_000) {
    float alpha = rand();
    loss = sqrt(alpha) * Y;
    if(loss < best) update();
}
```



Funded by the U.S. National Science Foundation and the Simons Foundation

# Code You Want to Write != Code You Want to Run

- Missing Algorithmic Optimizations
  - Example: expensive op factored out!
- Linear Algebra Optimizations
  - `diag(X) * Y + Z → X ⊙ slice(Y) + Z`
- Hardware optimizations
  - Rewrite to kernel which runs on devices 1, 2, 3, …, N, …

```
// Some example rules
 x + 0 -> x
 transpose(transpose(x)) -> x

// push slices up(reduce work)
 slice(add(a,b)) -> add(slice(a),slice(b))

// push pads down(reduce work)
 mul(pad(x,0),y) -> pad(mul(x,slice(y)),0)
```

```
x,y = tensor<10000xf32>
a = dot(x,y)
b = mul(a,z)
c = dot(b[0:10],4)
return c;
```

Funded by the U.S. National Science Foundation and the Simons Foundation

# Code You Want to Write != Code You Want to Run

- Missing Algorithmic Optimizations
  - Example: expensive op factored out!
- Linear Algebra Optimizations
  - `diag(X) * Y + Z → X ⊙ slice(Y) + Z`
- Hardware optimizations
  - Rewrite to kernel which runs on devices 1, 2, 3, …, N, …

```
// Some example rules
 x + 0 -> x
 transpose(transpose(x)) -> x

// push slices up(reduce work)
 slice(add(a,b)) -> add(slice(a),slice(b))

// push pads down(reduce work)
 mul(pad(x,0),y) -> pad(mul(x,slice(y)),0)
```

```
x,y = tensor<10000xf32>
a = dot(x,y)
b = mul(a,z)
c = dot(b[0:10],4)
return c;
```

```
x,y = tensor<10000xf32>
a = dot(x,y)
b = mul(a[0:10],z[0:10])
c = dot(b,4)
return c;
```

Funded by the U.S. National Science Foundation and the Simons Foundation

# Code You Want to Write != Code You Want to Run

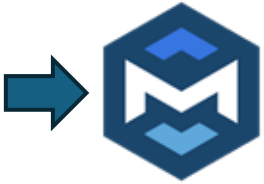
- Missing Algorithmic Optimizations
  - Example: expensive op factored out!
- Linear Algebra Optimizations
  - `diag(X) * Y + Z → X ⊙ slice(Y) + Z`
- Hardware optimizations
  - Rewrite to kernel which runs on devices 1, 2, 3, …, N, …

```
#define CEIL_DIV(M, N) (((M) + (N)-1) / (N))

template <const int BM, const int BN, const int BK, const int TM>
__global__ void sgemm1DBlocktiling(int M, int N, int K, float alpha,
                                   const float *A, const float *B, float beta,
                                   float *C) {
 const uint cRow = blockIdx.y;
 const uint cCol = blockIdx.x;

 const int threadCol = threadIdx.x % BN;
 const int threadRow = threadIdx.x / BN;

 // allocate space for the current blocktile in SMEM
 __shared__ float As[BM * BK];
 __shared__ float Bs[BK * BN];

 // Move blocktile to beginning of A's row and B's column
 A += cRow * BM * K;
 B += cCol * BN;
 C += cRow * BM * N + cCol * BN;

 // todo: adjust this to each thread to load multiple entries and
 // better exploit the cache sizes
 assert(BM * BK == blockDim.x);
 assert(BN * BK == blockDim.x);
 const uint innerColA = threadIdx.x % BK; // warp-level GMEM coalescing
 const uint innerRowA = threadIdx.x / BK;
 const uint innerColB = threadIdx.x % BN; // warp-level GMEM coalescing
 const uint innerRowB = threadIdx.x / BN;

 // allocate thread-local cache for results in registerfile
 float threadResults[TM] = {0.0};

 // outer loop over block tiles
 for (uint bkIdx = 0; bkIdx < K; bkIdx += BK) {
   // populate the SMEM caches
   As[innerRowA * BK + innerColA] = A[innerRowA * K + innerColA];
   Bs[innerRowB * BN + innerColB] = B[innerRowB * N + innerColB];
   __syncthreads();

   // advance blocktile
   A += BK;
   B += BK * N;

   ...
```

Funded by the U.S. National Science Foundation and the Simons Foundation

# Our Solution: Automatically Synthesize Efficient Code

Reactant Compiler
(C++, Julia, Python)



Generate MLIR

Bayesian Dialect
`enzyme.sample, …`
Domain specific operations & analyses

Lowering + Enzyme AD
+ Tensor Optimizations

Hardware-Specific Codegen
+ Optimization

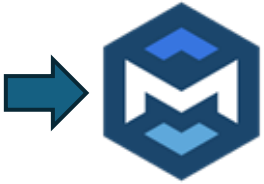
```
for i in axes(A, 1)
  tmp[i] = 0
  for j in axes(A, 2)
    tmp[i] += A[i, j] *
x[j]
  end
  for j in axes(A, 2)
    y[j] += A[i, j] *
tmp[i]
  end
end
```

Optimization passes

Funded by the U.S. National Science Foundation and the Simons Foundation

# Our Solution: Automatically Synthesize Efficient Code

Reactant Compiler
(C++, Julia, Python)



Bayesian Dialect
`enzyme.sample, …`
Domain specific operations & analyses

Generate MLIR

Lowering + Enzyme AD
+ Tensor Optimizations

Hardware-Specific Codegen
+ Optimization

```
for i in axes(A, 1)
  tmp[i] = 0
  for j in axes(A, 2)
    tmp[i] += A[i, j] *
x[j]
  end
  for j in axes(A, 2)
    y[j] += A[i, j] *
tmp[i]
  end
end
```

Optimization passes

Funded by the U.S. National Science Foundation and the Simons Foundation

# Reactant Performance Gains Outside Astronomy

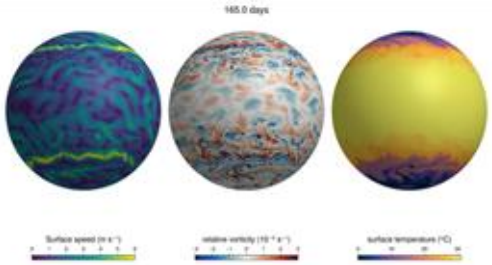
- Linear-algebra specific optimizations made a significant impact on training performance of a production LLM
  - 53% speedup of a run with 32x accelerators
  - 14-18.5% speedup for single accelerator

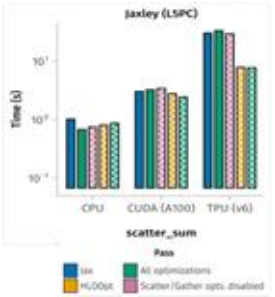
- Jaxley Biophysical Models Speedups (1.15x speedup on CPU, 1.33x speedup on A100, 3.92x speedup on TPU v6)

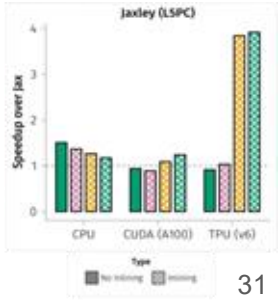
- Climate Model (31.8x on CPU, 4.4x on GPU)

```
Step: 2 loss: 12.880576133728027
Step: 3 loss: 12.785988807678223
Step: 4 loss: 12.652521133422852
Step: 5 loss: 12.482083320617676
…
Step: 19 loss: 9.190348625183105
Step: 20 loss: 8.928218841552734
Step: 21 loss: 8.660679817199707

Unopt: 0.479 samples/sec
```

```
…
Step: 19 loss: 9.189879417419434
Step: 20 loss: 8.929146766662598
Step: 21 loss: 8.659639358520508

Opt: 0.736 samples/sec
```



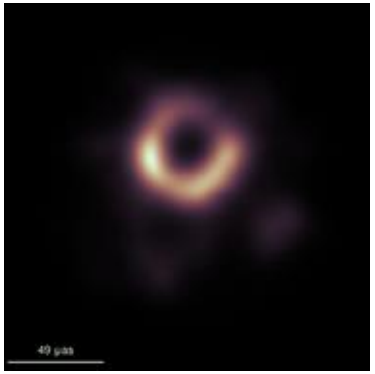


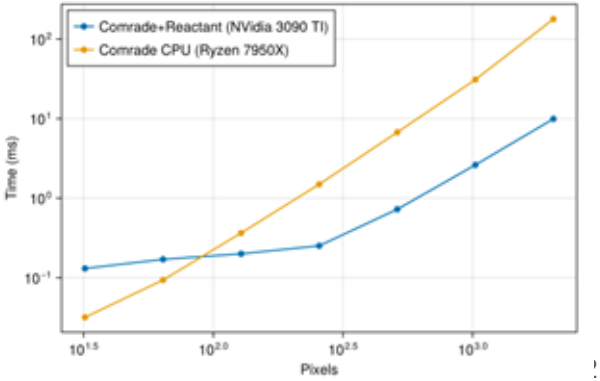

Funded by the U.S. National Science Foundation and the Simons Foundation

# Reactant Bayesian Inference on Astronomy Code

Made with Reactant + EnzymeMLIR




- End-to-end compilation of EHT imaging pipeline with Reactant, capable of reproducing original results
- Successful end-to-end execution on accelerators
- Preliminary analysis shows substantial improvements in speed for images with high resolution
  - >15x for 2k images on a single consumer GPU (NVIDIA 3090 @ Float64)

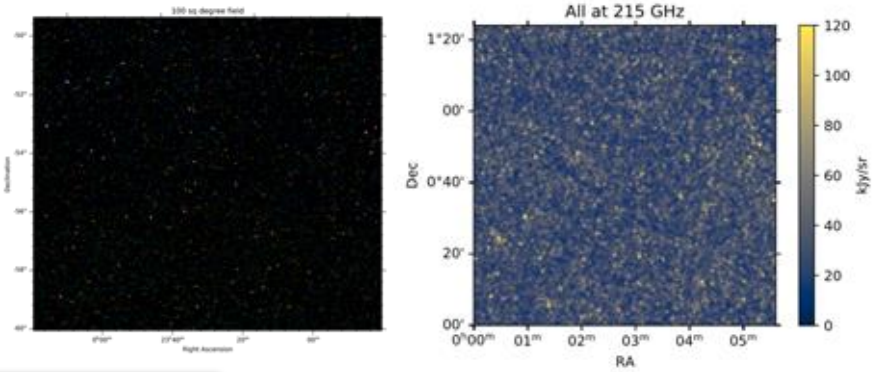


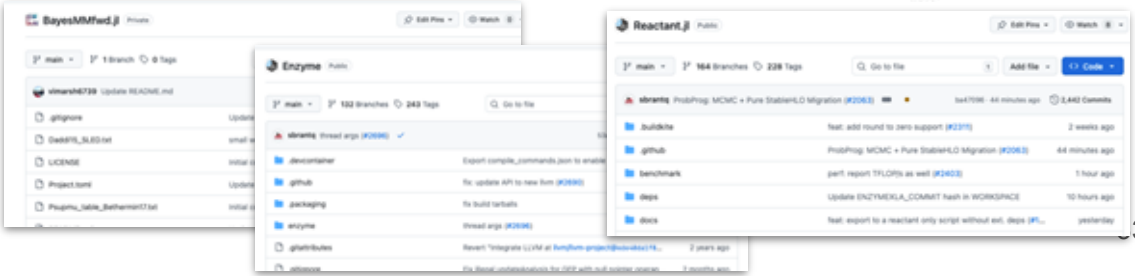

Funded by the U.S. National Science Foundation and the Simons Foundation

# End-to-End Status of the Project

- Successful end-to-end compilation and execution of forward model
- Successful end-to-end compilation and execution of gradient
- Reactant-compiled model beats prior workloads by an order of magnitude thanks to auto-parallelization and linear algebra transformations
- Integration of HMC in progress

- Generate mock galaxy catalog
- Add a Spectral Energy Distribution to each galaxy
- Calculate fluxes in each filter of interest




Funded by the U.S. National Science Foundation and the Simons Foundation

# Goal of this project

- Forward model [standalone publication]
  - Use dark matter halo models
  - Assign a Spectral Energy Distribution (SED) to each halo above a certain star formation rate
  - Simulate mock catalogs at different wavelengths (IR-mm)
  - Add radio emission to each source/ generate radio catalog
- Integration
  - Translate forward model code to julia
  - Make code compatible with Reactant
  - Translate source association bayesian inference code to julia
- Integration (contd.)
  - Check source association code compatibility with Reactant
- Reactant Compiler
  - Differentiation on Tensors
  - Linear Algebra Optimizations
  - Accelerator Backend - CPU, CUDA, TPU, ROCM
  - Bayesian Inference
    - Parallel Markov Chains
    - Sample-Invariant Code Motion
    - Dimensionality Reduction
    - Auto Block Decomposition
    - More Advanced Algorithms

SkAI
Funded by the U.S. National Science Foundation and the Simons Foundation

- Done
- In Progress
- To Do

# Success Metrics + Targets

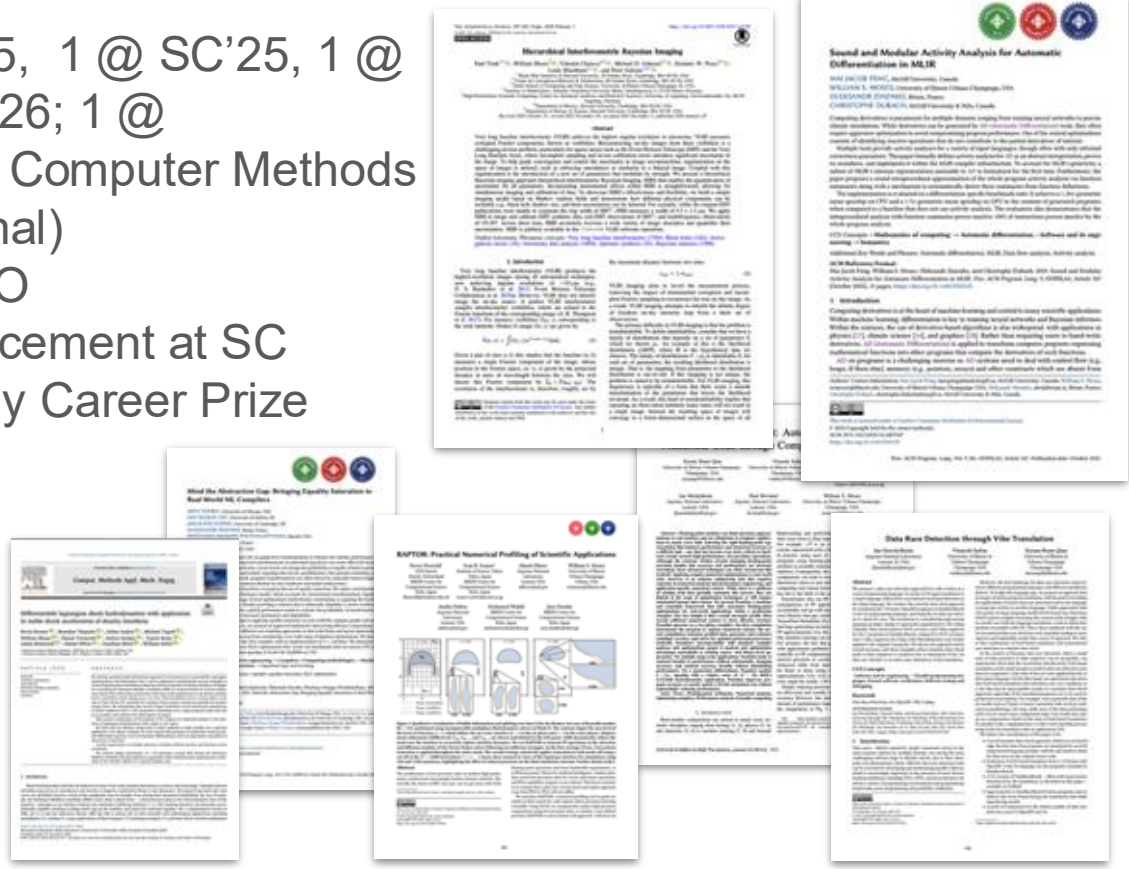
- Develop and release optimized forward model software to the community
  - Ability to generate mock catalogs at varying depths across the wavelengths and large area of sky
- Develop and release Bayesian inference source association code.
- Identify high-redshift sources in the full catalog as well as other interesting sources

- Implement automatic generation of efficient backward models
  - GPU acceleration for complex source associations
- Implement algorithmic, linear algebra, hardware optimizations
  - Scalable Bayesian inference on full catalogs with consumer-grade hardware

Funded by the U.S. National Science Foundation and the Simons Foundation

# Wins from the past six months

- 7 papers: (2 @ OOPSLA'25, 1 @ SC'25, 1 @ SCCorrectness, 1 @ CGO'26; 1 @ Astrophysical Journal, 1 @ Computer Methods in Applied Mechanics Journal)
- Distinguished Paper at CGO
  Best Reproducibility Advancement at SC
  SIAM Supercomputing Early Career Prize



SkAI

Funded by the U.S. National Science Foundation and the Simons Foundation

# Roadblocks and Challenges

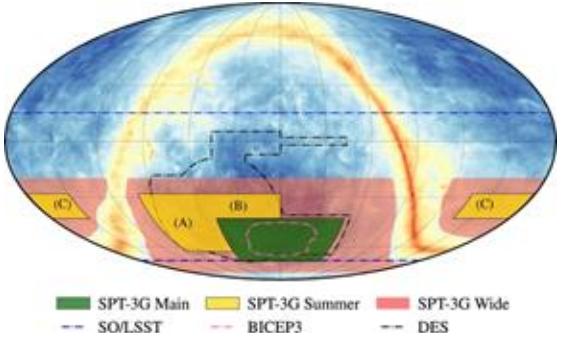
- No major significant theoretical challenges

- Biggest challenges mostly on software engineering:
  - More software engineering support is always helpful (and help us push things forward more quickly!)
  - Access to continuous integration infrastructure (in particular AMD GPUs)
  - Some of the distributed communication libraries nccl/rccl variants have strange nondeterministic timeouts at higher node counts (512 GPUs+)

- Advice on forward models related to different AGN types welcome.

SkAI
Funded by the U.S. National Science Foundation and the Simons Foundation

# Future scalability





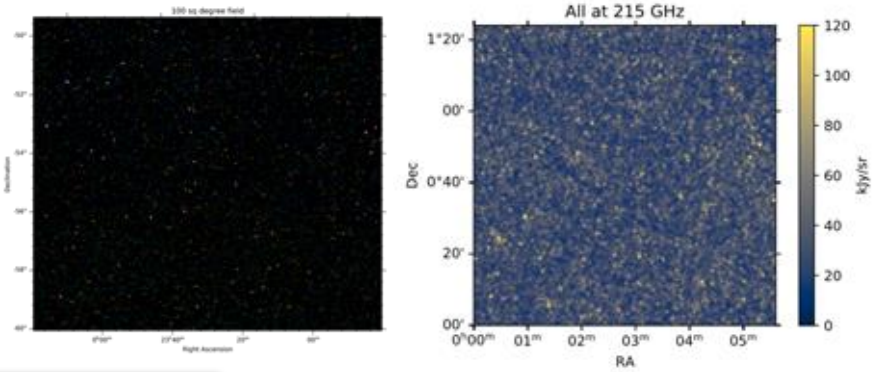
- Implement automatic Bayesian source associations for mm transients.
  - Need a model for mm emission mechanism for transient sources
  - This will be helped by building a dataset of simultaneously observed transients in millimeter and optical/IR.
- The fast Bayesian souce association code can be scaled to crossmatch/deblend sources in crowded fields. Rubin-LSST can benefit by combining datasets at multiple wavelengths using Bayesian inference (currently simple distance based matching is implemented in DP1).
- Support distributed computation (with automatic distribution algorithms), more advanced sampling techniques, and bayesian-specific optimizations

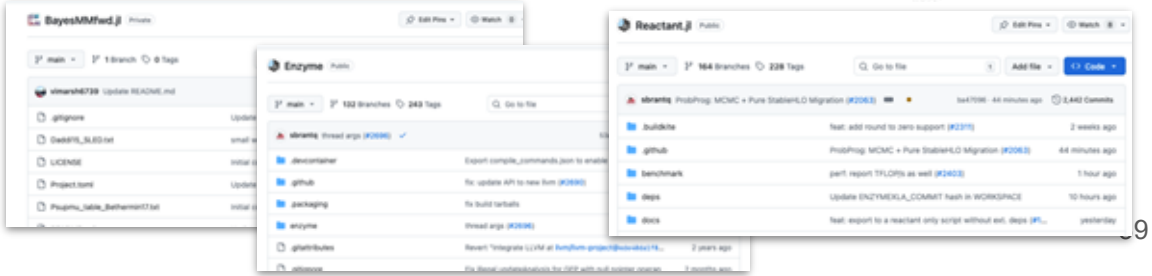
SkAI
Funded by the U.S. National Science Foundation and the Simons Foundation

# End-to-End Status of the Project

- Successful end-to-end compilation and execution of forward model
- Successful end-to-end compilation and execution of gradient
- Reactant-compiled model beats prior workloads by an order of magnitude thanks to auto-parallelization and linear algebra transformations
- Integration of HMC in progress

- Generate mock galaxy catalog
- Add a Spectral Energy Distribution to each galaxy
- Calculate fluxes in each filter of interest



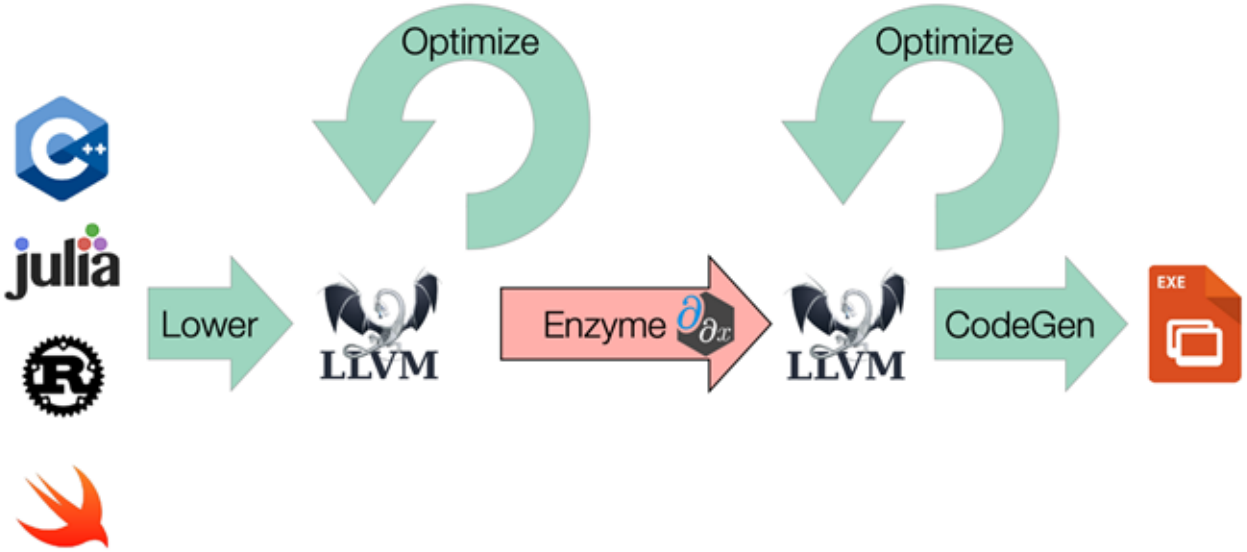

Funded by the U.S. National Science Foundation and the Simons Foundation

# Backward slide

- Forward model …
- We want efficient backward models that compute parameters given observations
- This requires exploring a high-dimensional parameters space efficiently

# Automatic Generation of Backward Model

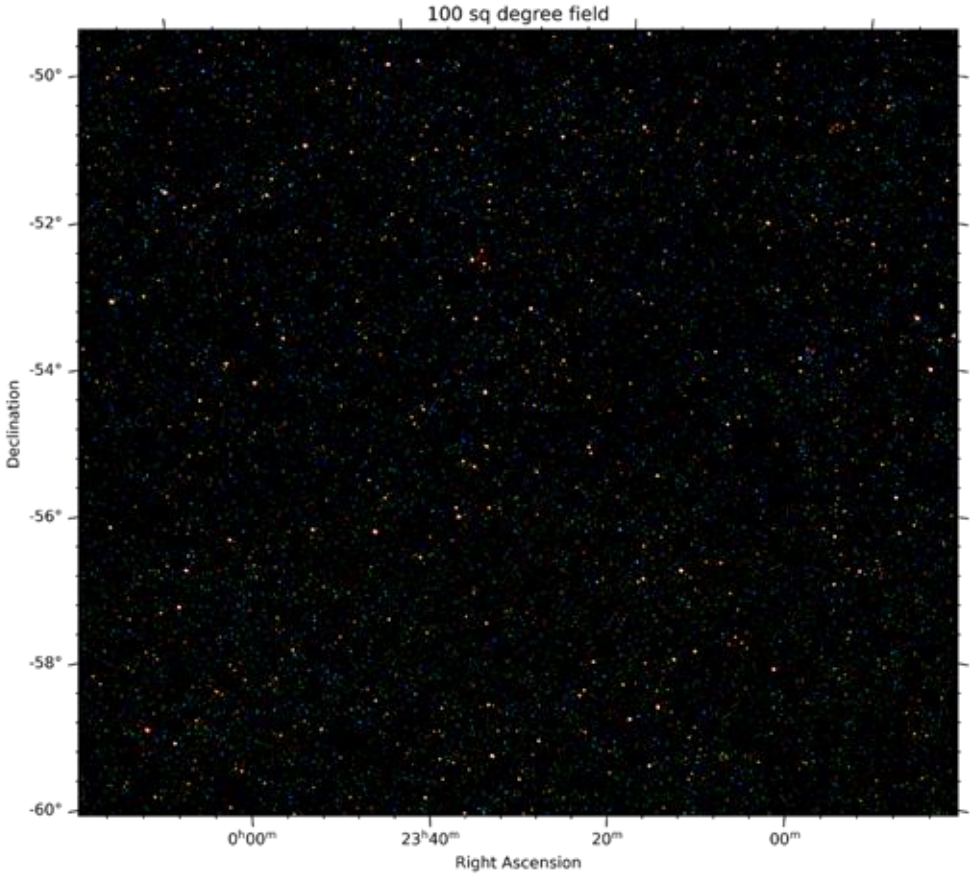
- Goal: Build first-class probabilistic programming with solvers in Reactant
- **Reactant**: an alternative Julia compiler into MLIR that delivers domain-specific optimizations and GPU/TPU acceleration
- No-U-Turn-Sampler available on GitHub: [EnzymeAD/Reactant.jl]



## 100 deg$^2$ of SPT-3G sources

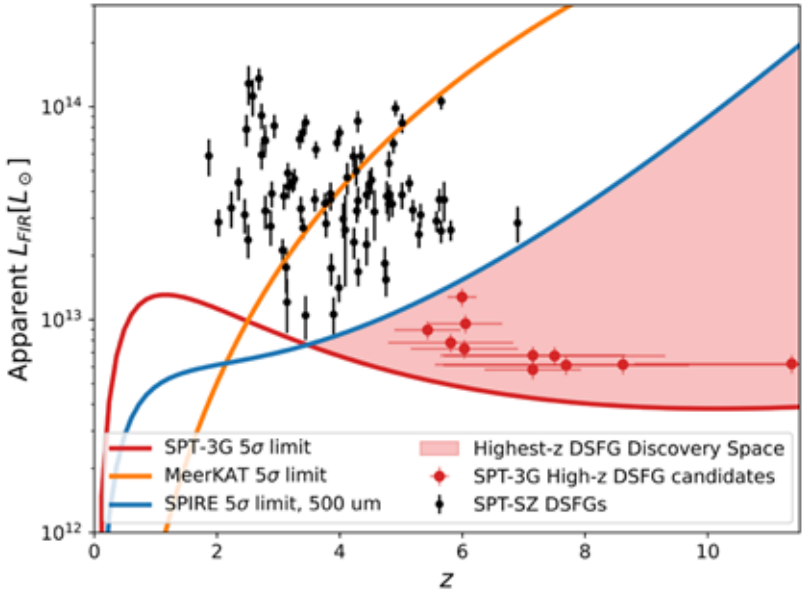
**95 GHz**
**150 GHz**
**220 GHz**

- 2k SPT sources.
- 200k Herschel (FIR) sources.
- 225k MeerKAT (radio) sources.



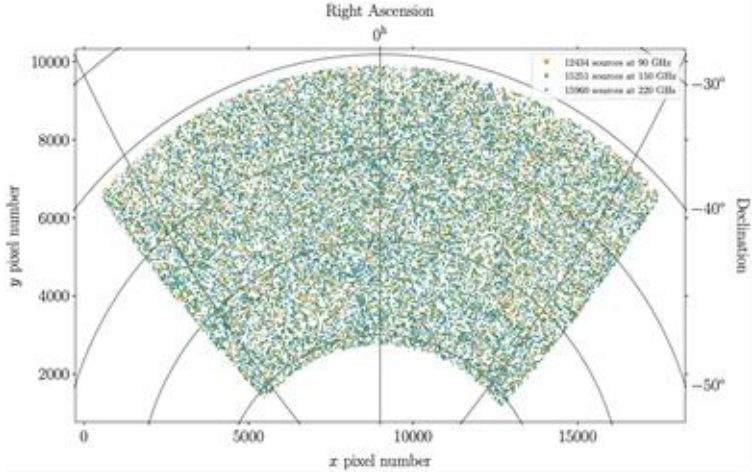
# Search for the most distant dusty star forming galaxy

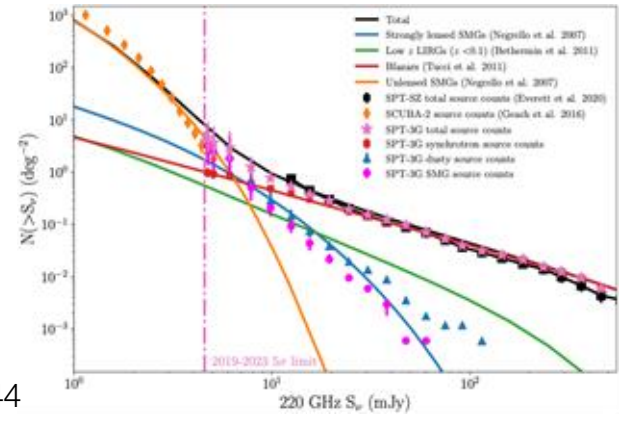
- 78 candidate sources found to have no other matches
- 11 very high redshift candidates selected after visual inspection
- ALMA program approved for spectroscopic confirmation



# Astro+AI Gameplan

- Part 1: First 20 months
- Automate Bayesian inference for source associations [Astro+AI]
- What we need?
  - Forward models amenable to differentiable and probabilistic programming
- How?
  - We can use empirical source population models to create mock mm-wave catalogs and associate a SED to each source.
  - Train forward model using these mock catalogs.



# AI Gameplan

- Natively combining Differentiation with Optimization has been quite successful
  - Automatically generate backwards pass, previously hand-written
  - **Performance improvements enable new science**

# AI Gameplan

- Natively combining Differentiation with Optimization has been quite successful
  - Automatically generate backwards pass, previously hand-written
  - Performance improvements enable new science

- Automating solvers (i.e. bayesian inference) is more complicated
  - Uses differentiation for continuous variables, but our solver must also handle discrete decisions and higher-level algorithms
  - **Goal: Build first-class probabilistic programming with solver into the compiler (enabling optimization, discrete choices, hybrid neural+classical solvers, etc)**

# AI Methods & Approaches

**Methods & Approaches (5-7 minutes *AI side only*)**

- **What methods are being used?**
  Focus on the AI or computational techniques applied (e.g., machine learning models, natural language processing, simulations).

- **Why these methods?**
  Briefly explain why these approaches are suitable for the problem and how they compare to other approaches.

- **If not commonly used: how do they work at a high level?**
  Avoid technical jargon—use intuitive explanations or diagrams to illustrate the process. *Make sure you define any acronyms!*

- With regard to any AI models in it would be great if speakers could note specifically:
  - What the inputs/outputs are,
  - What the usual or baseline approach is,
  - Where that breaks down,
  - And how the developed approach improves on that.

SkAI
Funded by the U.S. National Science Foundation and the Simons Foundation

# Status of the Project

**Current Status & Results (5 minutes)**

- **What is the current status?**
  Share preliminary results or insights, emphasizing what is novel or has been unexpected. This is not meant to necessarily include all work that has been performed. Where is the project at this moment?

- **What are the project's goals and success metrics?**
  Define what success looks like—scientific breakthroughs, practical applications, etc.

- **What are the next research targets?**
  Outline the upcoming phases of the project and what you hope to learn or achieve.

Funded by the U.S. National Science Foundation and the Simons Foundation

# Roadblocks and Challenges

**Challenges (10 minutes, designed to spur discussion/additional connection across SkAI)**

- **What challenges have emerged?**
  Discuss technical, conceptual, or collaborative hurdles—especially those that arise in astro-AI interdisciplinary space.

- **Where could you use advice?**
  SkAI has a wide array of expertise and experience. Please use this opportunity to request input and make connections where they can benefit the project.

- **Do you see any new opportunities unfolding?**
  Highlight new opportunities, unexpected applications, or broader impacts.

Funded by the U.S. National Science Foundation and the Simons Foundation

Reactant Compiler
(C++, Julia, Python)

Bayesian Dialect
`enzyme.sample, …`
Domain specific operations & analyses

Generate MLIR

```
for i in axes(A, 1)
  tmp[i] = 0
  for j in axes(A, 2)
    tmp[i] += A[i, j] *
x[j]
  end
  for j in axes(A, 2)
    y[j] += A[i, j] *
tmp[i]
  end
end
```

Lowering + Enzyme AD
+ Tensor Optimizations

Hardware-Specific Codegen
+ Optimization