chunking

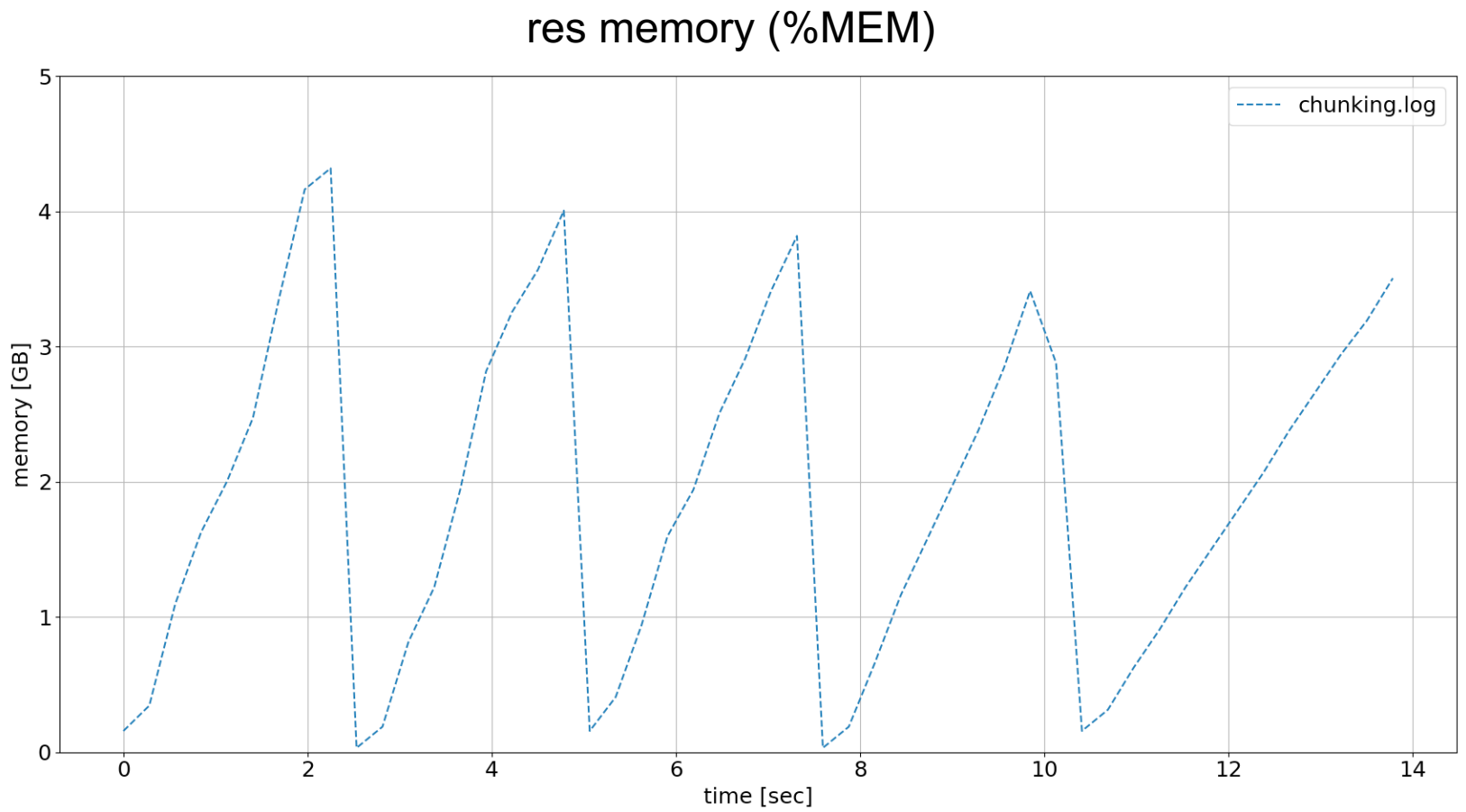
pytorch/src/DNNROIFinding.cxx

```
iframe (tags) -> Eigen -> Eigen (downsample) -> Tensor ->

torch.chunk(nchunk, dim)
loop chunks {
  torch::jit::IValue -> ITensorSet -> (inferencing) ITensorSet -> Tensor
}
torch.cat{{}, dim}

-> Eigen -> Eigen (upsample)
```

Memory for 1, 2, 4, 8,100 chunking

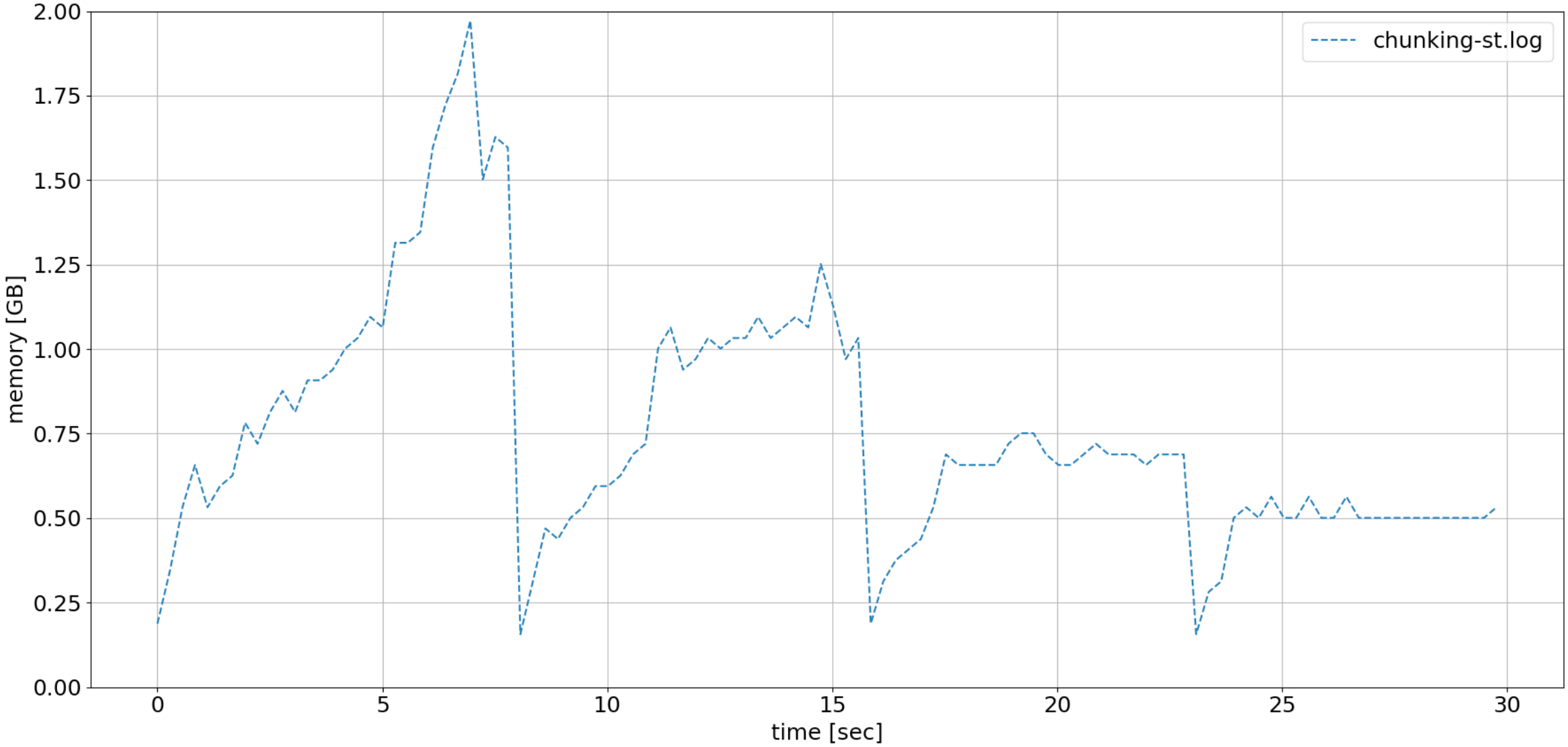
ExeMon:

```
 16 Memory: MEM: total: size=1.10656e+07K, res=5.06988e+06K increment: size=8.70751e+06K, res=4.87017e+06K inference
 35 Memory: MEM: total: size=9.09029e+06K, res=4.45606e+06K increment: size=6.73219e+06K, res=4.25552e+06K inference
 58 Memory: MEM: total: size=8.05338e+06K, res=4.10013e+06K increment: size=5.69553e+06K, res=3.89966e+06K inference
 89 Memory: MEM: total: size=7.57813e+06K, res=3.96266e+06K increment: size=5.22004e+06K, res=3.763e+06K inference
304 Memory: MEM: total: size=7.01704e+06K, res=3.71398e+06K increment: size=4.65894e+06K, res=3.51409e+06K inference
```



Brookhaven National Laboratory

chunking 1, 2, 4, 8, `torch::NoGradGuard no_grad;`



Brookhaven National Laboratory

Gray's fix + no_grad

- sbndgpvm04
- ref: WCT 0.27.1
- new: Gray's fix + ~~no_grad~~ (BV already fixed it!)
- chunking
- found a bug

```
35  void Pytorch::TorchService::configure(const WireCell::Configuration& cfg)
36  {
37      auto dev = get<std::string>(cfg, "device", "cpu");
38      m_ctx.connect(dev);
39
40      auto model_path = Persist::resolve(cfg["model"].asString());
41      if (model_path.empty()) {
42          log->critical("no TorchScript model file provided");
43          THROW(ValueError() << errmsg{"no TorchScript model file provided"});
44      }
45
46      // Use almost 1/2 the memory and 3/4 the time.
47      torch::NoGradGuard no_grad;
```

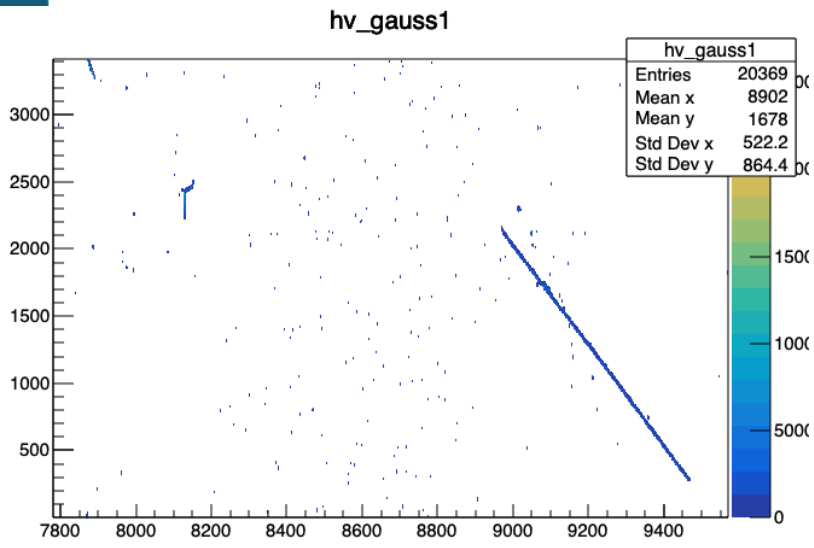
	Total time, sec	OSP time, sec	dnn time, sec	VmHWM, GB
ref, nodnn, 6000ticks	28			1.4
ref, dnn	113		19*2=38	7.7
new, dnn	108		19*2=38	7.8
new, dnn, 3500	70		10*2=20	5.2
new, dnn, 3500, 4chunks	80		~21	2.4
new, dnn, 3500, 8chunks				1.6
new, dnn, 3500, 4chunks MP rebin 10	75			2.0

~47sec in preparing additional output in OSP?

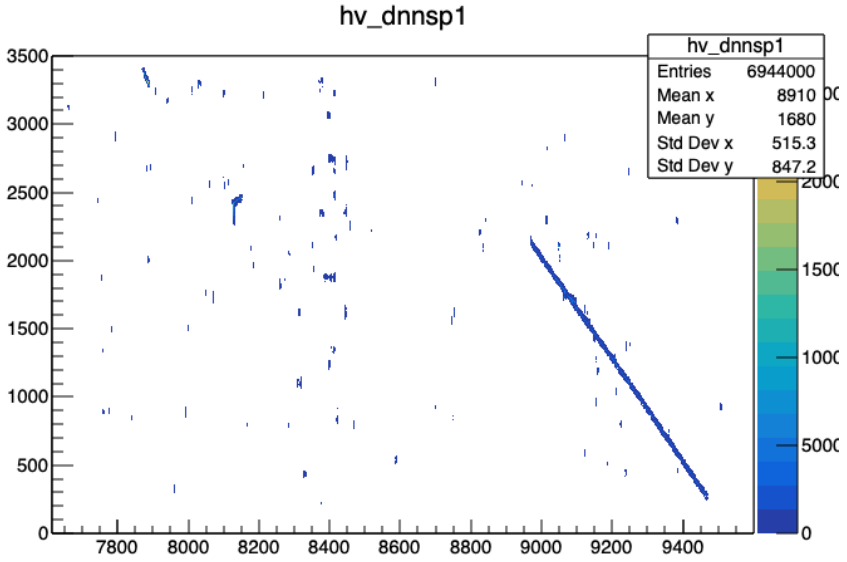
- reduce OSP resolution

Still bugs to fix

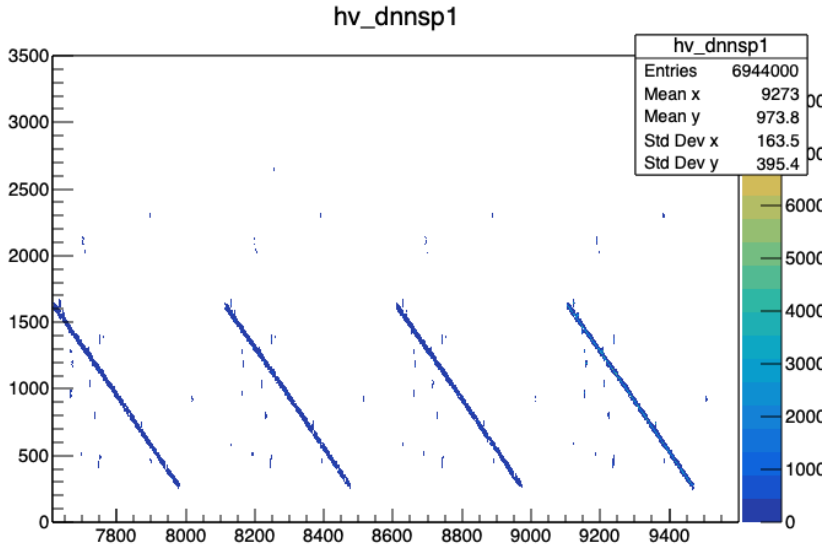
ref-nodnn-6000ticks



new-3500



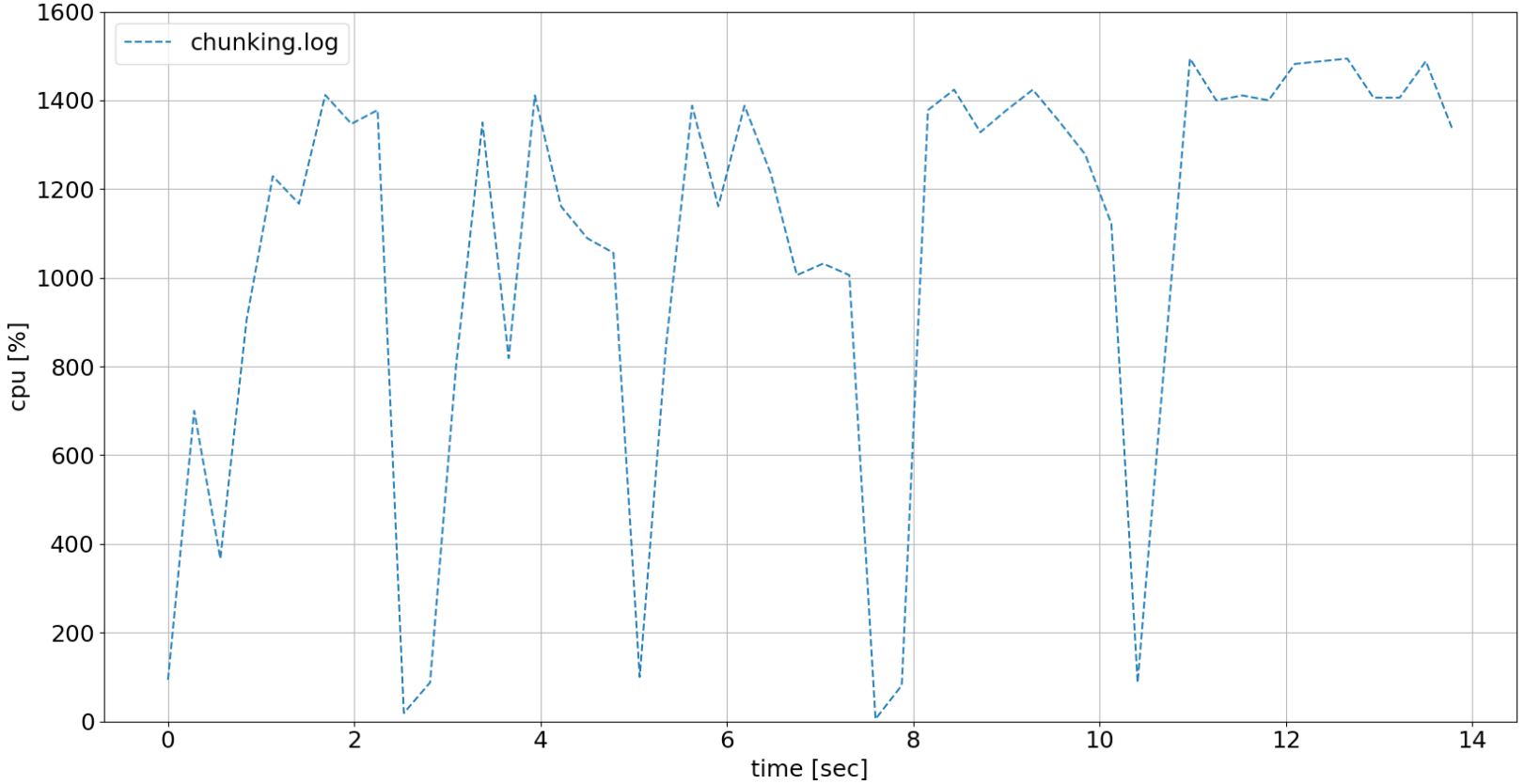
new-3500-4chunks



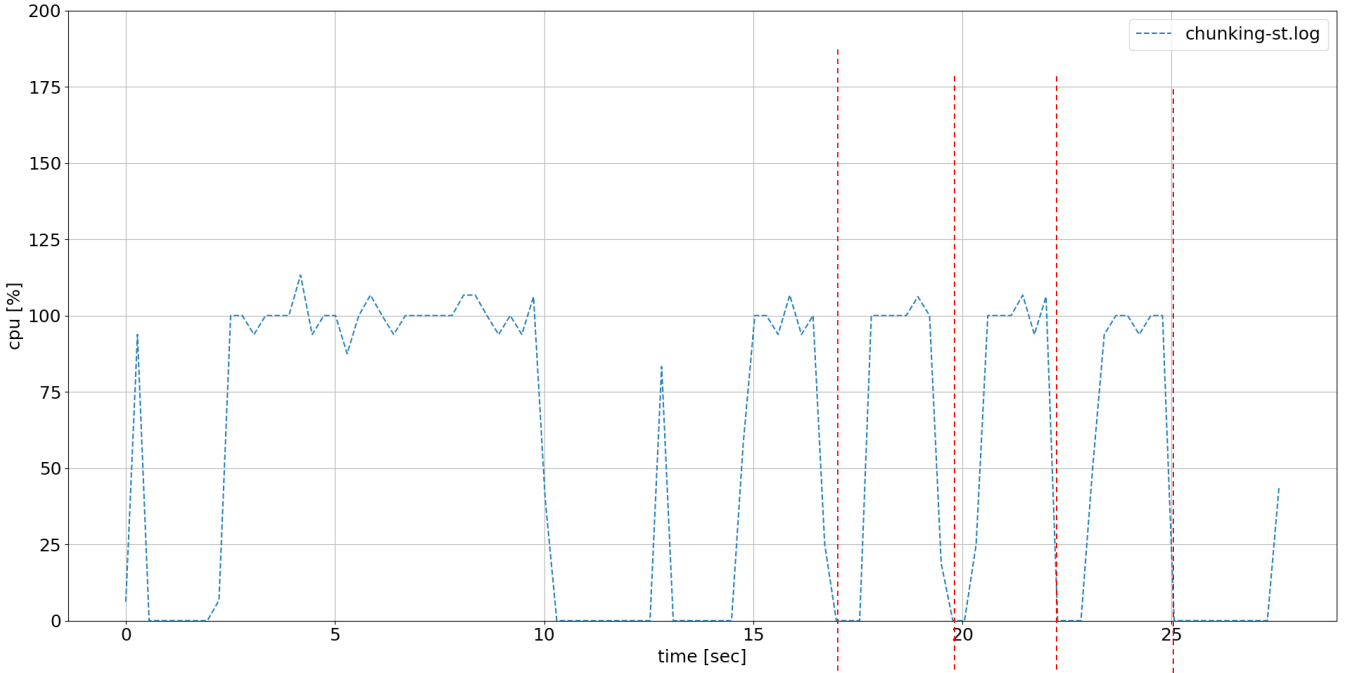
Brookhaven National Laboratory

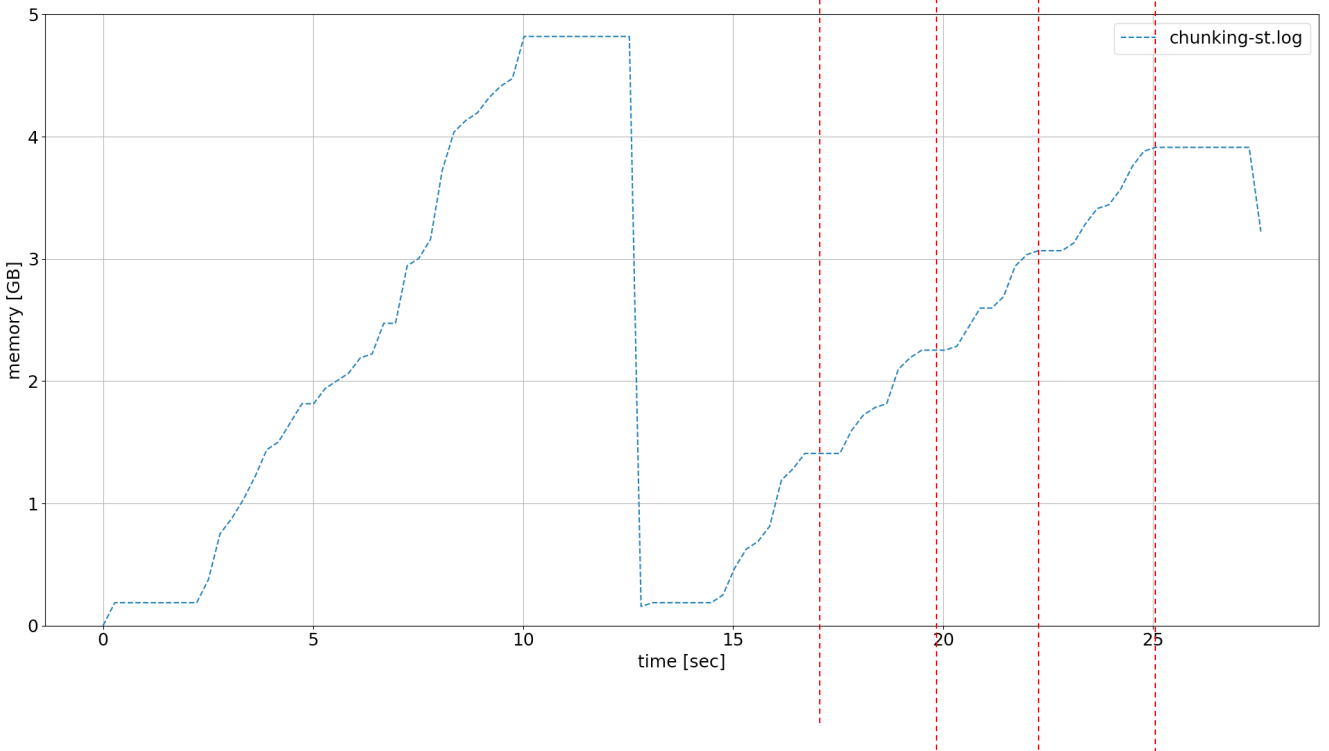
backups

Memory for 1, 2, 4, 8,100 chunking



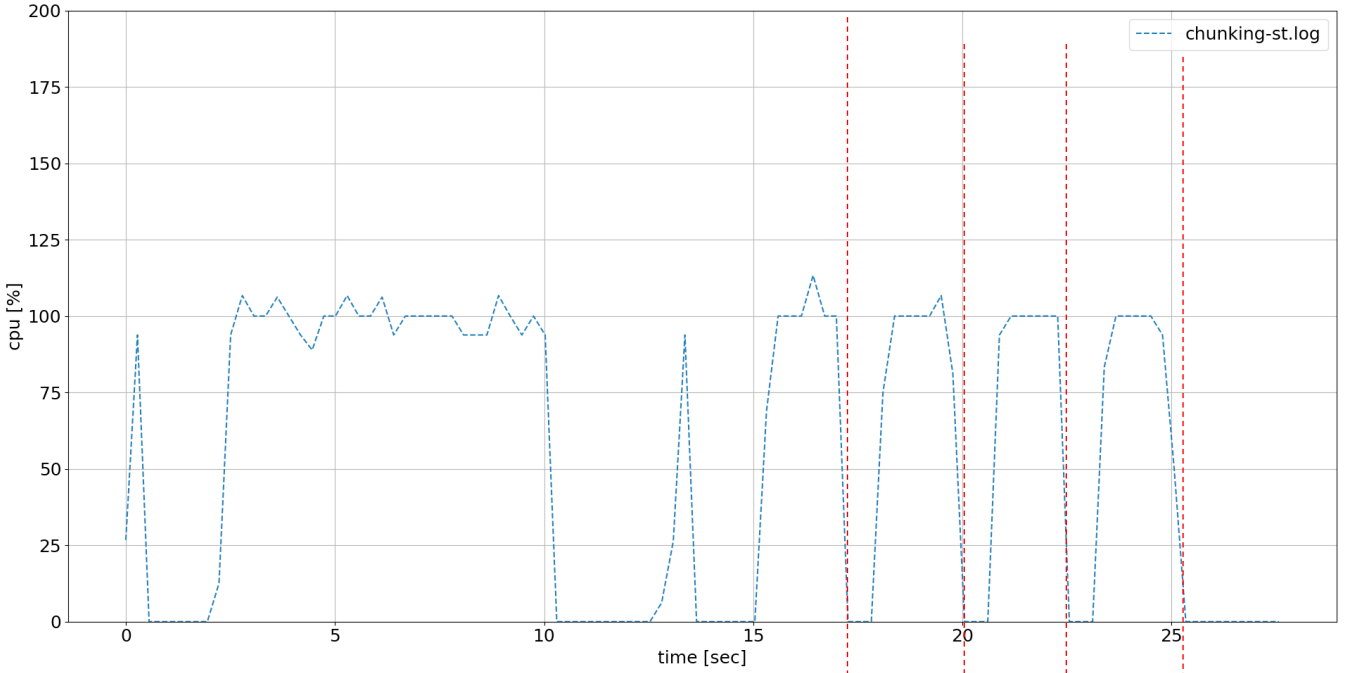
orig

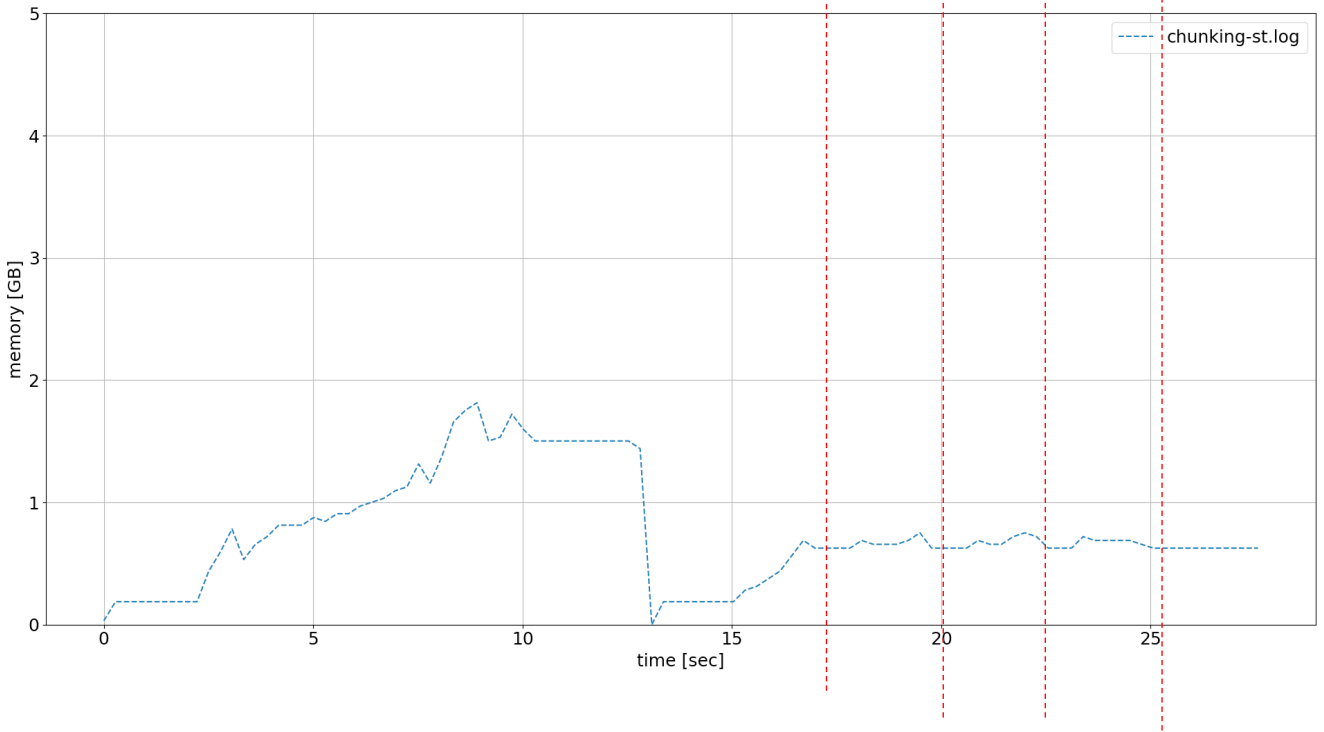




Brookhaven National Laboratory

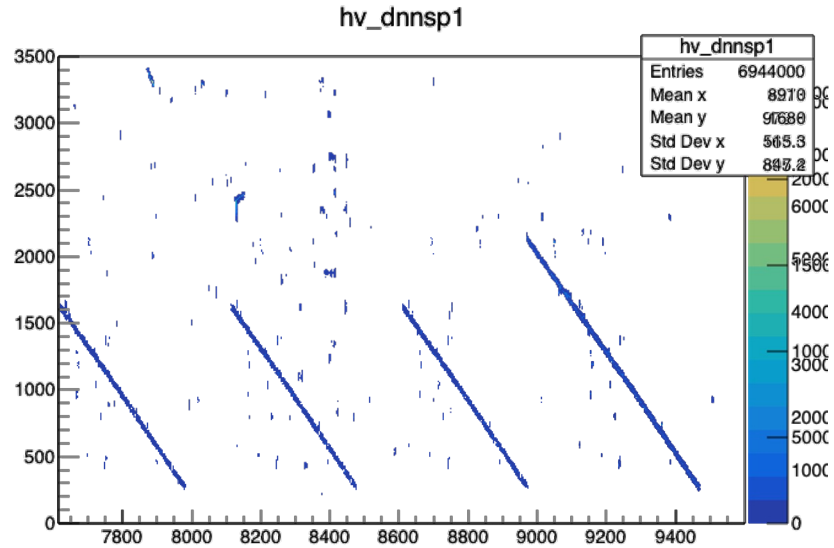
```
torch::NoGradGuard no_grad;
```

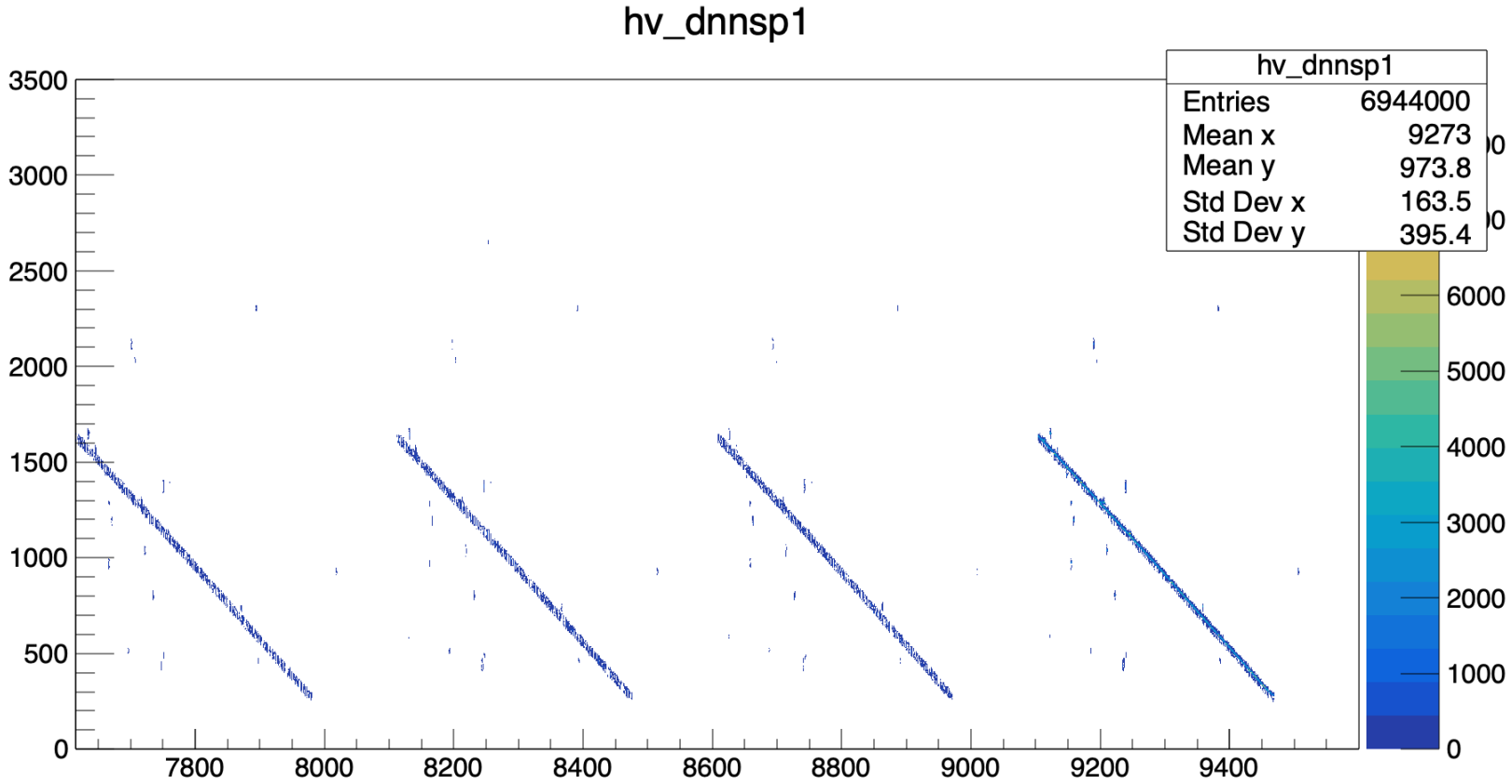


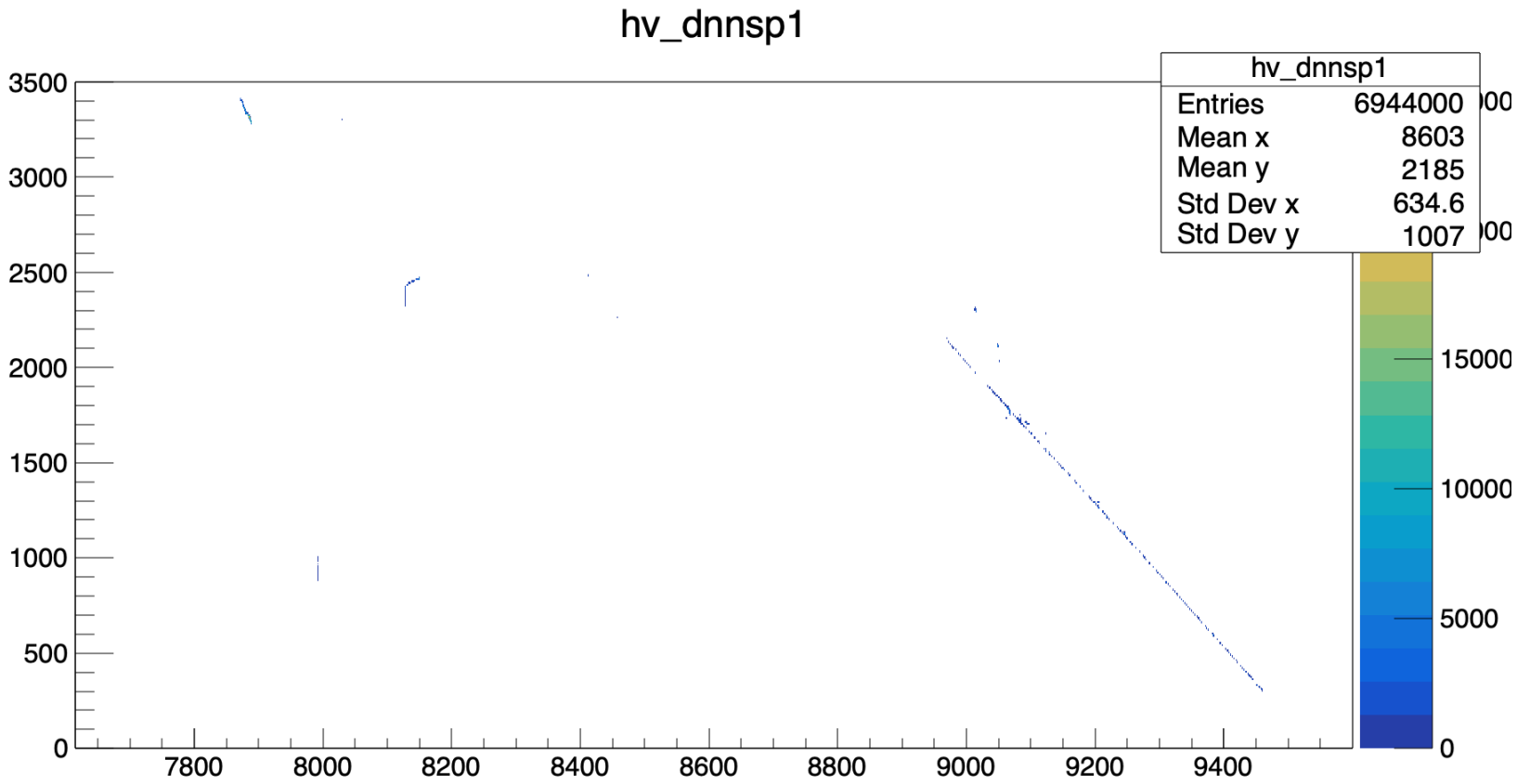


Brookhaven National Laboratory

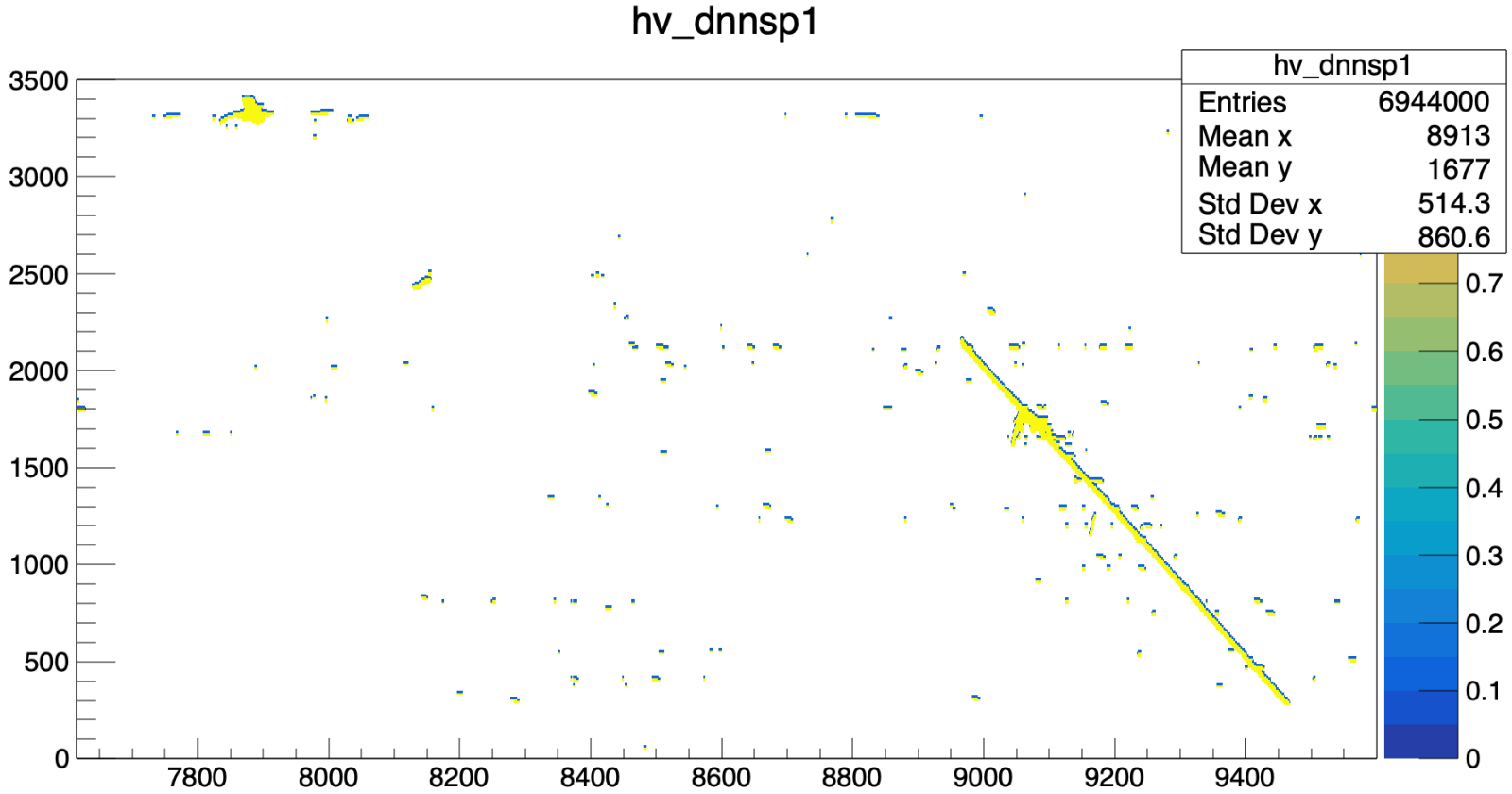
eigen after scaling: vector of 3 of 1984x340
chunking: n x 3 x 1984/n x 340


hv_dnnsp1
hv_dnnsp1
Entries 6944000
Mean x
Mean y
Std Dev x
Std Dev y
3500
3000
2500
2000
1500
1000
500
0
7800
8000
8200
8400
8600
8800
9000
9200
9400


hv_dnnsp1
hv_dnnsp1
Entries 6944000
Mean x 9273
Mean y 973.8
Std Dev x 163.5
Std Dev y 395.4
3500
3000
2500
2000
1500
1000
500
0
7800
8000
8200
8400
8600
8800
9000
9200
9400
6000
5000
4000
3000
2000
1000
0



Brookhaven National Laboratory



Brookhaven National Laboratory

```
for (auto chunk : chunks) {
    std::cout << "chunk size: " << chunk.sizes() << std::endl;
    std::vector<torch::IValue> itens {chunk};
    auto iitens = Pytorch::to_itensor(itens);
    auto oitens = m_forward->forward(iitens);
    torch::Tensor ochunk = Pytorch::from_itensor({oitens}).front().toTensor().cpu();
    std::cout << "ochunk size: " << ochunk.sizes() << std::endl;
    outputs.push_back(ochunk.clone());
```

```cpp
std::vector<torch::IValue> Pytorch::from_itensor(const ITensorSet::pointer &inputs, const bool gpu)
{
    std::vector<torch::IValue> ret;

    for (auto iten : *inputs->tensors()) {
        if (iten->shape().size() != 4) {
            THROW(ValueError() << errmsg{"iten->shape().size()!=4"});
        }
        // TODO determine data type from metadata
        auto ten = torch::from_blob((float *) iten->data(), {(long) iten->shape()[0], (long) iten->shape()[1],
                                                             (long) iten->shape()[2], (long) iten->shape()[3]});
        if (gpu) {
            ten = ten.to(torch::Device(torch::kCUDA, 0));
            assert(ten.device().type() == torch::kCUDA);
            ret.push_back(ten);
            //ret.push_back(ten.cuda());
        }
        else {
            ret.push_back(ten);
        }
    }

    return ret;
}
```

from_blob does not copy

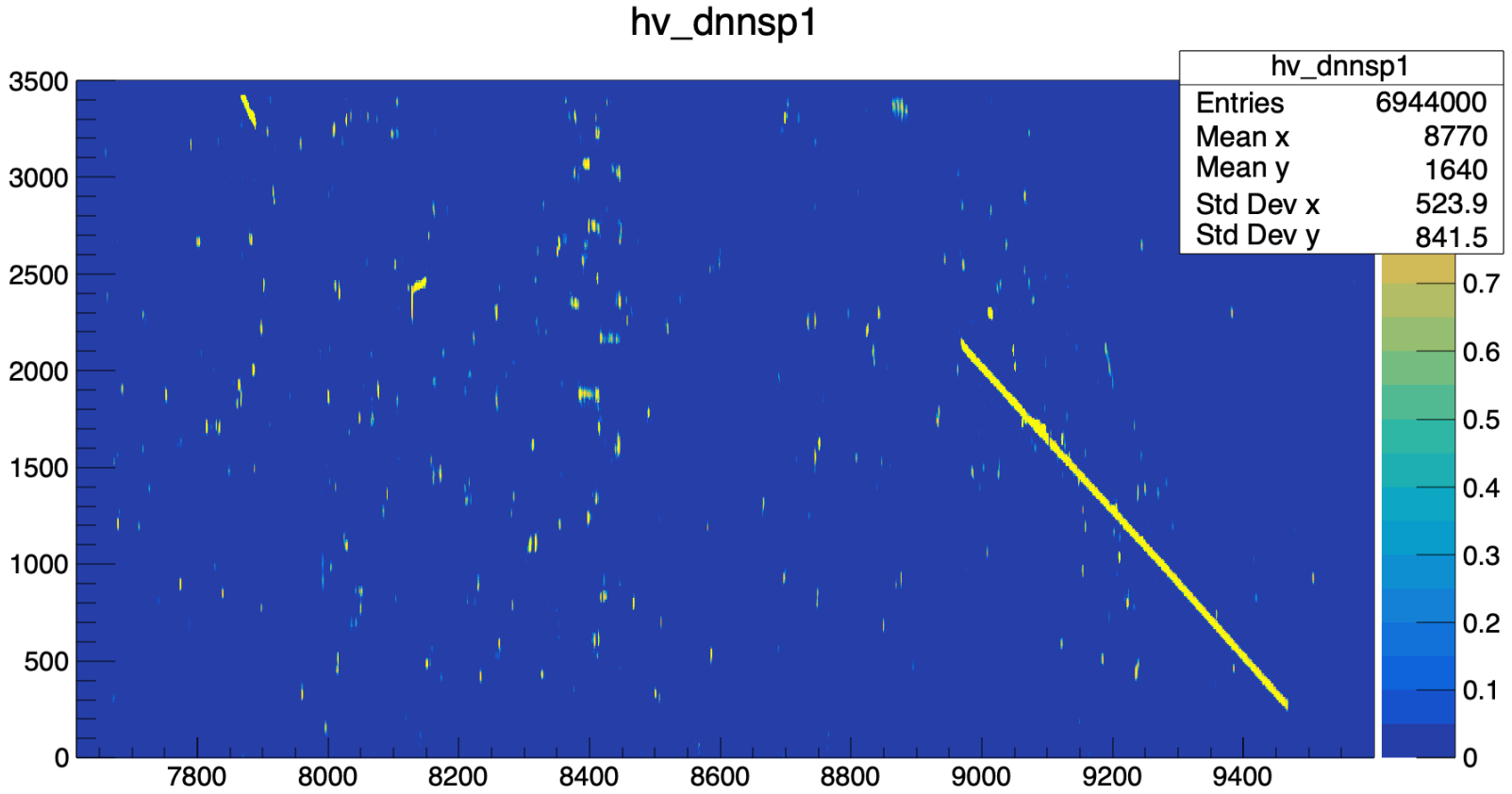
Brookhaven National Laboratory

```cpp
ITensorSet::pointer Pytorch::to_itensor(const std::vector<torch::IValue> &inputs)
{
    ITensor::vector *itv = new ITensor::vector;

    int ind = 0;
    for (auto ival : inputs) {
        auto ten = ival.toTensor().cpu();
        if (ten.dim() != 4) {
            THROW(ValueError() << errmsg{"ten.dim() != 4"});
        }
        std::vector<size_t> shape = {(size_t) ten.size(0), (size_t) ten.size(1), (size_t) ten.size(2),
                                     (size_t) ten.size(3)};
        // TODO need to figure out type from dtyp
        Aux::SimpleTensor *st = new Aux::SimpleTensor(shape, (float*)nullptr);
        size_t nbyte = 4;
        for (auto n : shape) nbyte *= n;
        auto data = (float *) st->data();
        std::cout << "tgt: " << data << std::endl;
        std::cout << "src: " << (float *) ten[0][0].data<float>() << std::endl;
        memcpy(data, (float *) ten[0][0].data<float>(), nbyte);
        itv->push_back(ITensor::pointer(st));
        ++ind;
    }

    // FIXME use a correct seqno
    int seqno = 0;
    Configuration md;

    return std::make_shared<Aux::SimpleTensorSet>(seqno, md, ITensor::shared_vector(itv));
}
```

```
chunk size: [1, 3, 496, 350]
tgt: 0x1fcb6fc0
src: 0x2b0d0c80
[13:32:11.003] D [ torch  ] <T
tgt: 0x25b91c80
src: 0x2013aec0
ochunk size: [1, 1, 496, 350]
chunk size: [1, 3, 496, 350]
tgt: 0x1fcb6fc0
src: 0x2b17a500
[13:32:13.848] D [ torch  ] <T
tgt: 0x25b91c80
src: 0x1d7b7500
ochunk size: [1, 1, 496, 350]
chunk size: [1, 3, 496, 350]
tgt: 0x1fcb6fc0
src: 0x2b223d80
[13:32:16.718] D [ torch  ] <T
tgt: 0x25b91c80
src: 0x1c9aa540
ochunk size: [1, 1, 496, 350]
chunk size: [1, 3, 496, 350]
tgt: 0x1fcb6fc0
src: 0x2b2cd600
[13:32:19.455] D [ torch  ] <T
tgt: 0x25b91c80
src: 0x1c418500
```

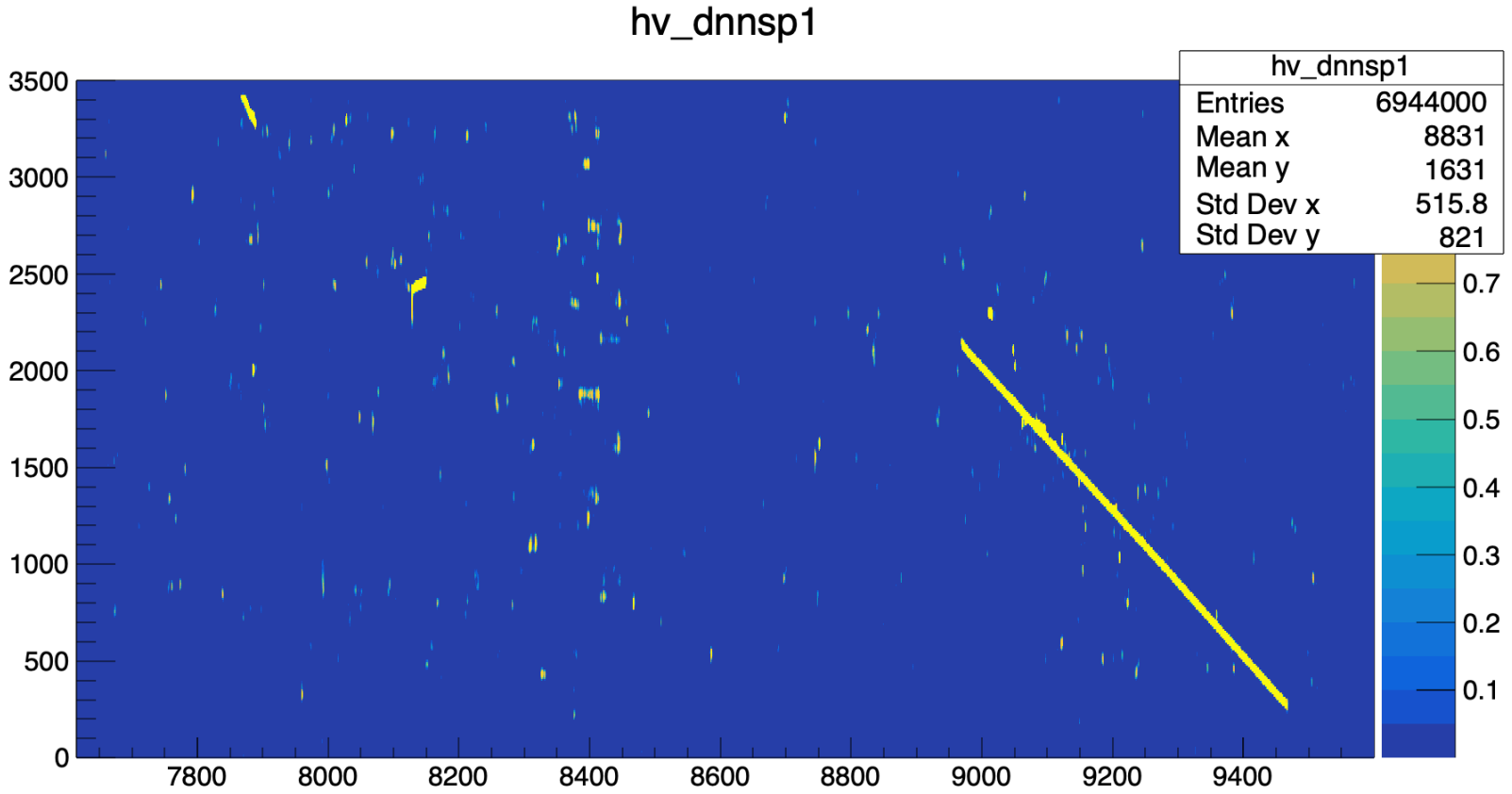
Brookhaven National Laboratory

4 chunks



Brookhaven National Laboratory

1 chunk


hv_dnnsp1
hv_dnnsp1
Entries 6944000
Mean x 8831
Mean y 1631
Std Dev x 515.8
Std Dev y 821
3500
3000
2500
2000
1500
1000
500
0
7800
8000
8200
8400
8600
8800
9000
9200
9400
0.7
0.6
0.5
0.4
0.3
0.2
0.1

	Total time, sec	NV	OSP time, sec	dnn time, sec	VmHWM, GB
no-dnn	26.6	~4.5*2	~8*2		1.4
chunk-1	75.8	~4.5*2	~9.5*2	~11*4	5.1
chunk-4	72.6	~4.5*2	~9.5*2	~11*4	2.2