The HPC Challenge Benchmarks and the PMaC project

- Certificates of relevance for benchmarks
  - Do they cover a useful performance space?
  - Do they enable reasoning about expected app. Performance?
- How *practically* to measure memory access patterns in nature
- Useful performance taxonomy

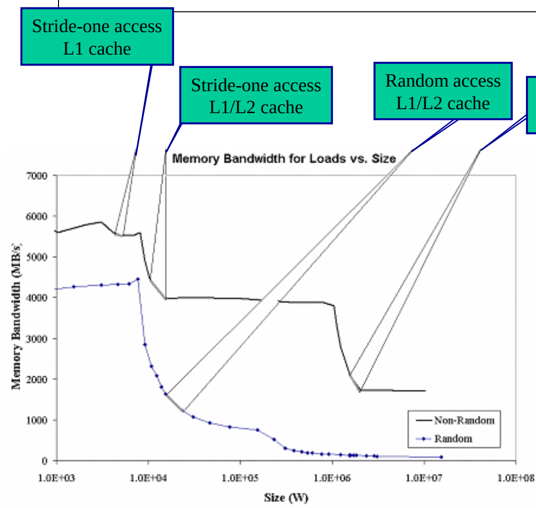
Components of a Performance Prediction Framework

- *Machine Profile* - characterizations of the rates at which a machine can (or is projected to) carry out fundamental operations abstract from the particular application
- *Application Signature* - detailed summaries of the fundamental operations to be carried out by the application independent of any particular machine

Combine Machine Profile and Application Signature using:

- *Convolution Method* - algebraic mapping of the application signature onto the machine profile to calculate a performance prediction

MAPS Data



MAPS curve for TCSini for random and non-random loads.

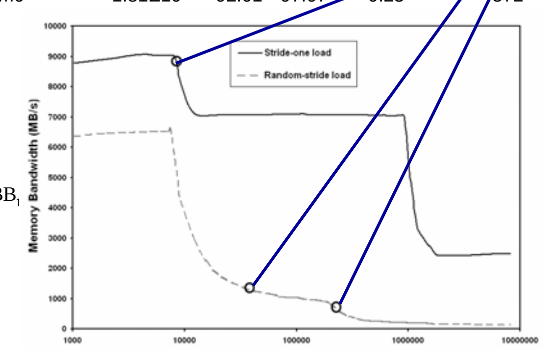
MAPS –
Memory bandwidth benchmark measures memory rates (MB/s) for different levels of cache (L1, L2, L3, Main Memory) and different access patterns (stride-one and random)

Convolutions

MetaSim trace collected on Cobalt$_{60}$ simulating SC45 memory structure

Basic-Block Number	Procedure Name	# Memory References	L1 hit rate	L2 hit rate	Random ratio	Memory Bandwidth
5247	Walldst	2.22E11	97.28	99.99	0.00	8851
10729	Poorgrd	4.90E08	88.97	92.29	0.20	1327
8649	Ucm6	1.81E10	92.01	97.07	0.23	572

$$\text{Memory time} = \sum_{i=1}^{n} \text{MemOpsBBi}/\text{MemRateBB}_1$$



Results-Predictions for AVUS (Cobalt60)

AVUS TI-05 standard data set on 64 CPUs

System	Actual time (s)	Predicted time (s)	% Error
NAVO IBM PWR3 (Habu)	8,601	11,180	+30%
ARL IBM PWR3 (Brainerd)	10,675	10,385	-3%
MHPCC IBM PWR3 (Tempest)	8,354	9,488	+14%
MHPCC IBM PWR4 (Hurricane)	4,932	4,258	-14%
NAVO IBM PWR4 (Marcellus)	4,375	4,445	+2%
ARL IBM PWR4 (Shelton)		6,192	
NAVO IBM PWR4+ (Romulus)	3,272	3,239	-1%
ASC HP SC45	3,334	2,688	-19%
ARL Linux Networx Xeon Cluster		3,459	

PERC

Spatial and Temporal Locality

How could one Quantify the Spatial and Temporal Locality in a Real Code?

$$\text{SpatialScore}(N) = \sum_{i=1}^{N} (\text{Refs Stride } i / i) / \text{Total Refs}$$

$$\text{TemporalScore}(N) = \text{Observed Reuse} / (\text{Total Refs} - \text{Spatial Refs})$$

PERC

It’s Harder Than it Looks

Where does one plot RandomAccess?

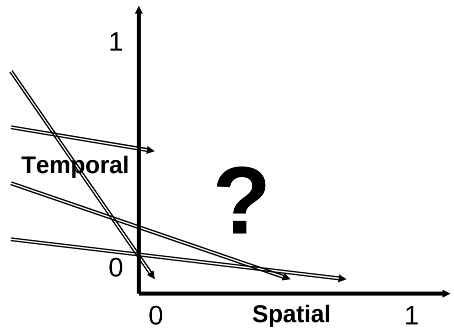
```
for ( i = 0; i < N; i++) {
  add = random_number;
  table[add] ^= random_number;
}
```

Update (design goal)

Load + Store (temporal)

Load + Store (spatial)

Two loads + Store



HPC Challenge Benchmarks on axes of spatial and temporal locality