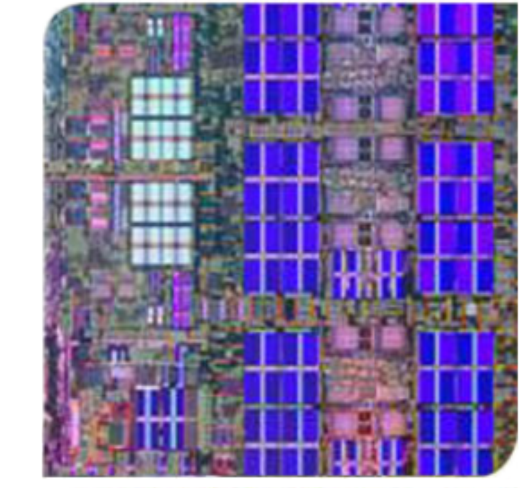
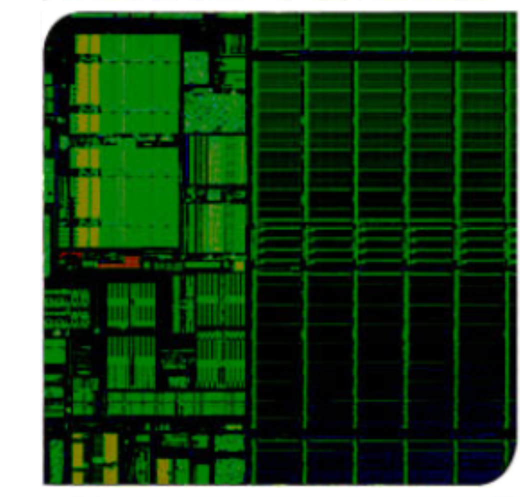
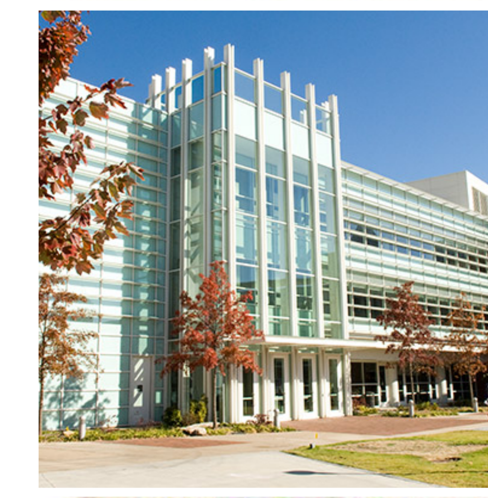
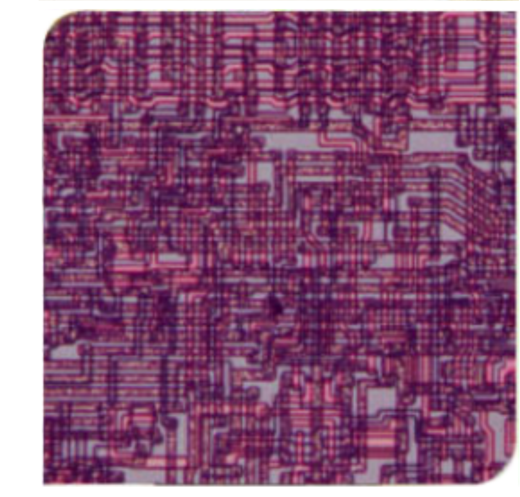
Creating Robust Deep Neural Networks With Coded Distributed Computing for IoT Systems

Ramyad Hadidi and Hyesoon Kim
Georgia Institute of Technology

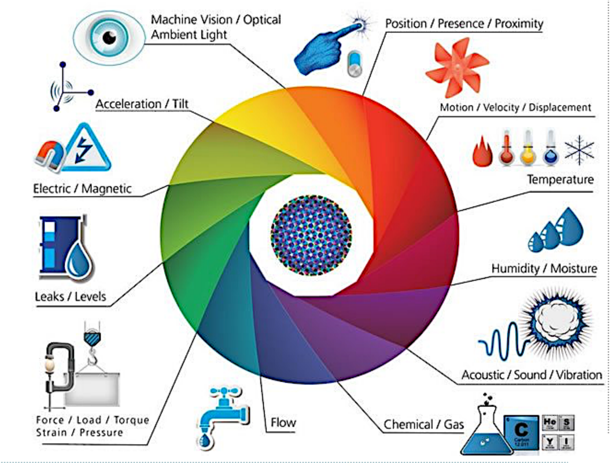





Internet of Things Devices

- Internet of Things (**IoT**) devices
  - Have access to an abundance of raw data
  - In home, work, or vehicle

IoT: Raw Data & Processing

- IoT is gaining ground with the widespread of
  - Embedded processors
  - Ubiquitous wireless networks
- Access to raw data
  - Understand it!
  - Real-time constraints
  - Limited resources
    - Power
    - Compute

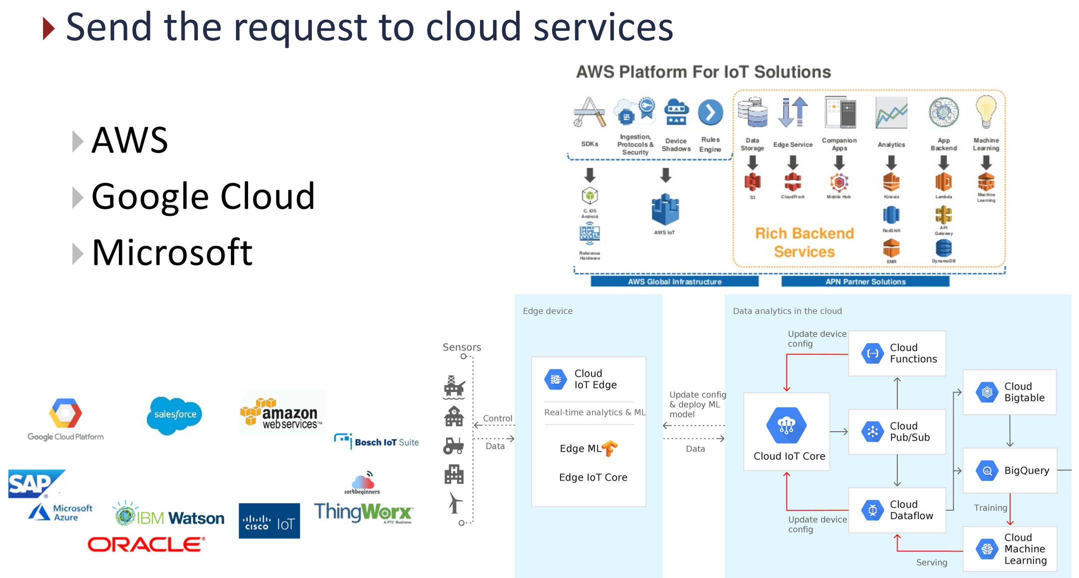


IoT: DNN-based Processing

- With deep neural networks (**DNNs**):
  - With DNNs IoTs can
    - Process several new data types and
    - Understand behaviors
  - Speech, vision, video, and text
- But, DNNs are resource hungry
  - Cannot met real-time constraints on IoT devices
  - Several DNNs cannot be executed on IoTs

Approach 1: Offload to Cloud

- Send the request to cloud services
  - AWS
  - Google Cloud
  - Microsoft

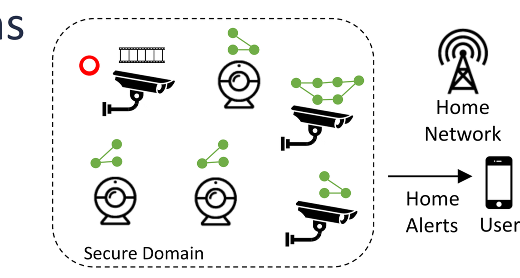


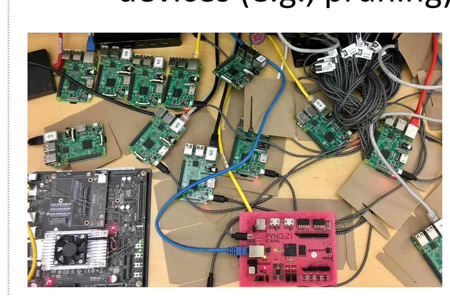
Why Cloud is not Always a Solution

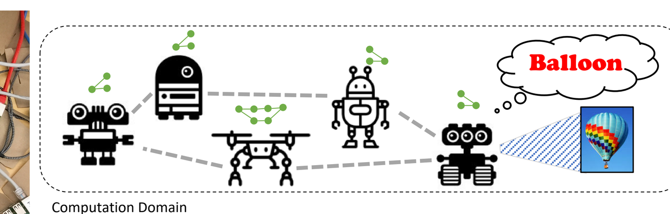
- Unreliable connections to the cloud
  - Plus low bandwidth and high latency
- Disconnected Devices
- **Privacy**
  - Privacy preserving learning (e.g., differential privacy)
  - Privacy preserving inference (e.g. homomorphic encryption)
- Personalization
- Federated learning

Approach 2: IoT Collaboration

- Distribute computations with collaboration
  - To meet demands of DNNs
  - On top of common DNN techniques for constrained devices (e.g., pruning)







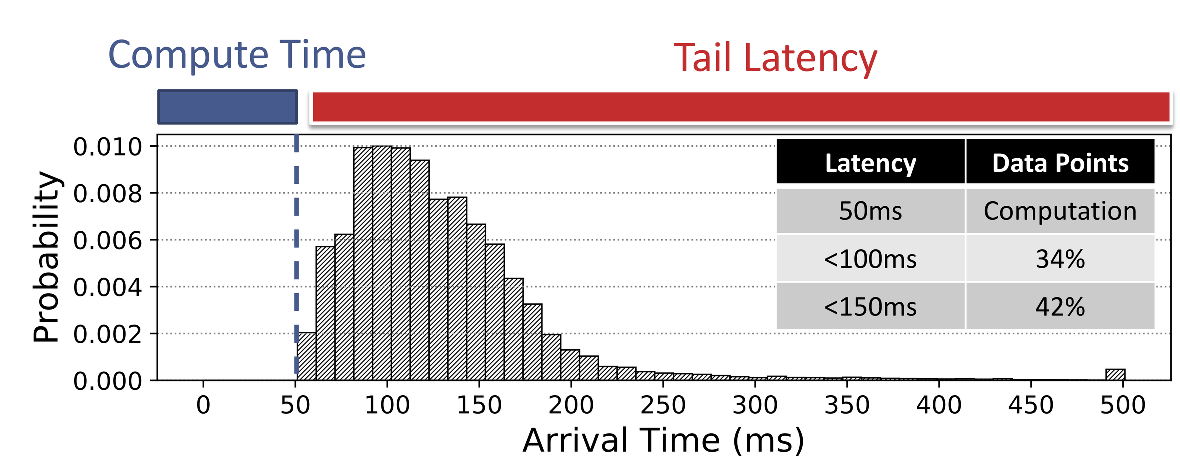
IoT Collaboration Pros & Cons

- Assuming DNN performance barrier is solved with collaboration among IoT devices

Pros	Cons
Not Dependent on Cloud	Unreliable Latencies
Privacy Preserving	Accuracy Drop due to Data Loss & Device Failure
Enables Personalized Insight	

Challenges Impact: Unreliable Latencies

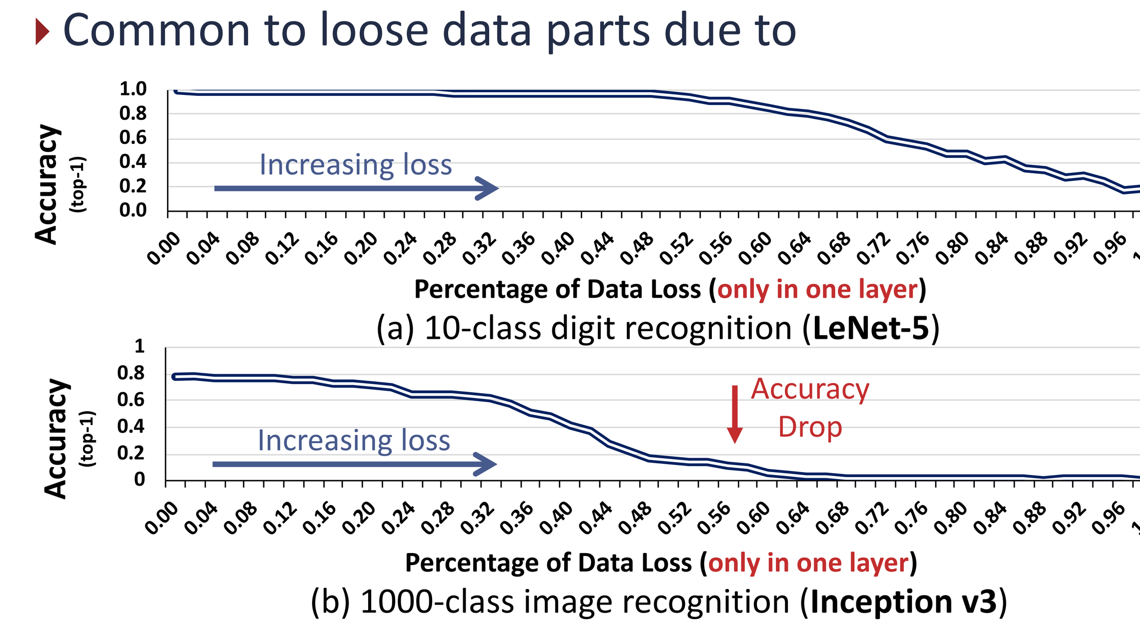
- Histogram of arrival times in 4-node system performing AlexNet (model parallelism).



Long Tail and Max Latency -> Straggler Problem

Challenges Impact: Accuracy Drop

- Common to loose data parts due to

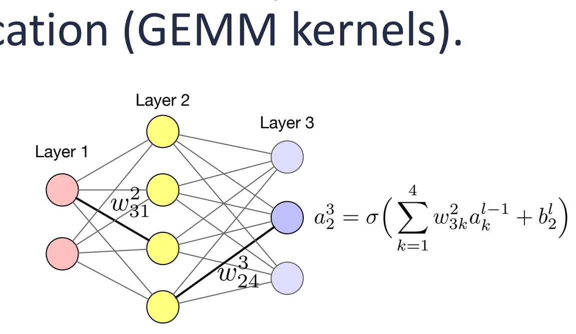


High Accuracy Drop

Computation of DNNs

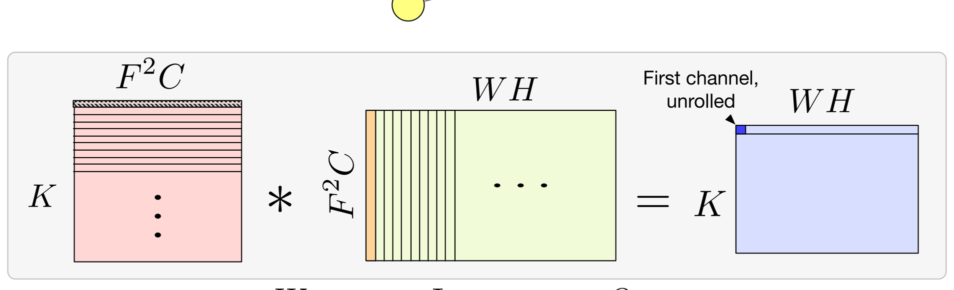
- Each layer's computations can be represented as matrix-matrix multiplication (GEMM kernels).

Fully-connected layer:



$$a_2^3 = \sigma\left(\sum_{k=1}^{4} w_{2k}^3 a_k^{l-1} + b_2^l\right)$$

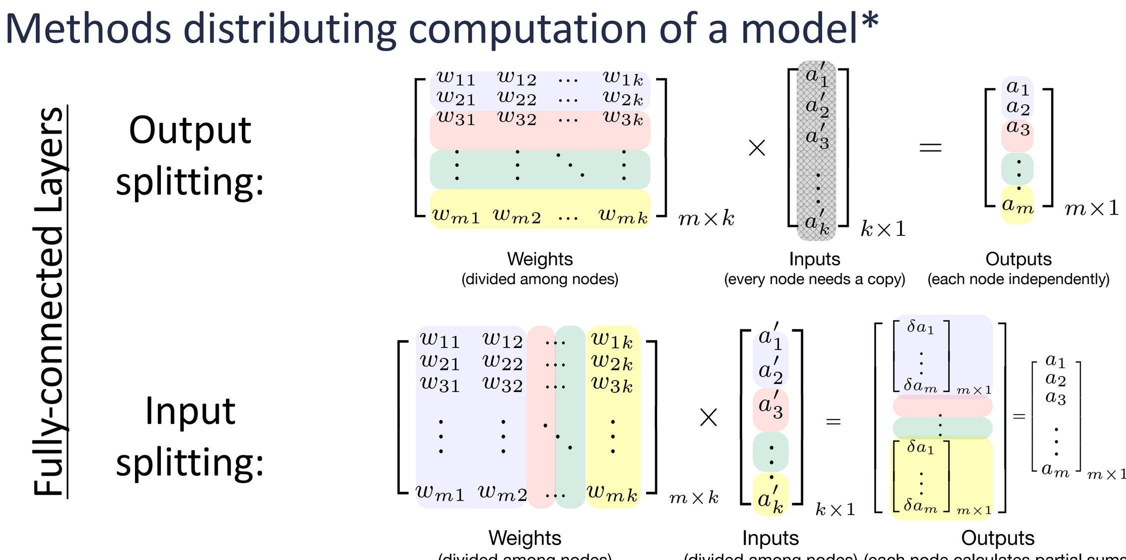
Conv. layer:



$$W_{K\times F^2C} \times I_{F^2C\times WH} = O_{K\times WH}$$

Computation Distribution of DNNs

- Methods distributing computation of a model*



- Same can be applied on conv. layers*
  - Channel , spatial , and filter splitting

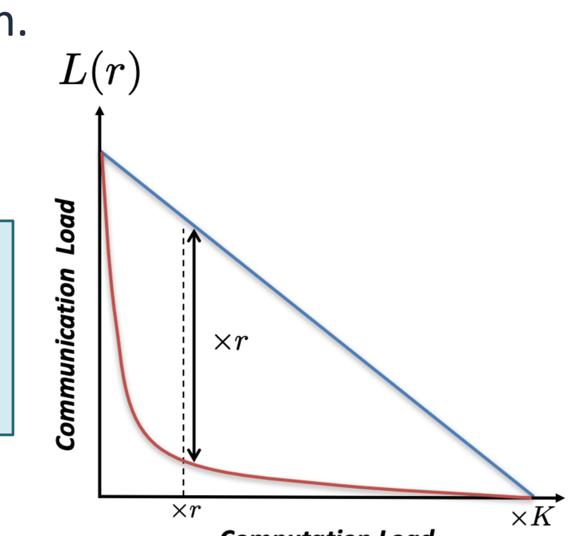
Hadidi, Ramyad, et al. "Towards collaborative inferencing of deep neural networks on internet of things devices." *IEEE Internet of Things Journal* (2020).

Coded Distributed Computing (CDC)

- Designed for MapReduce workloads (2018)*
- Preforming redundant or coded computer per node to reduce communication.

This work: **DNNs on IoT**

More Compute / Node = More Reliability



* Li, Songze, et al. "A fundamental tradeoff between computation and communication in distributed computing." *IEEE Transactions on Information Theory* 64.1 (2018): 109-128.

Using CDC for Robustness

- Add column-wise summation of the weights:

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{11}+w_{21} & w_{12}+w_{22} \end{bmatrix} \times \begin{bmatrix} a'_1 \\ a'_2 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_1+a_2 \end{bmatrix}$$

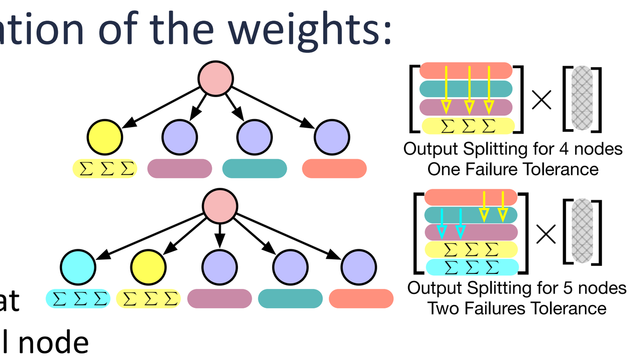
- The new weights are constant, so done in offline

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{:1}^{cdc} & w_{:2}^{cdc} \end{bmatrix} \times \begin{bmatrix} a'_1 \\ a'_2 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a^{cdc} \end{bmatrix}$$

- Distribute outputs among nodes
  - Thus, applicable only to output-splitting methods

How to Distribute CDC and Recover?

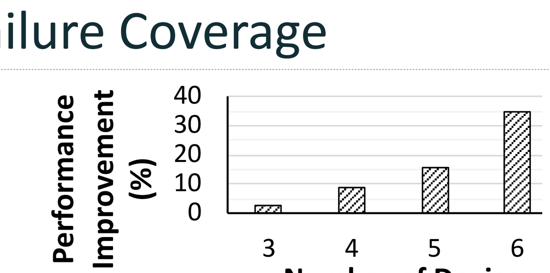
- Add column-wise summation of the weights:
  - Simple example (one output/device)
  - Recovery
    - **Subtraction vs. Multiplication**
    - You also needs the weights, that you would not have in the final node
  - Multiple out/device: Just create a new weight matrix



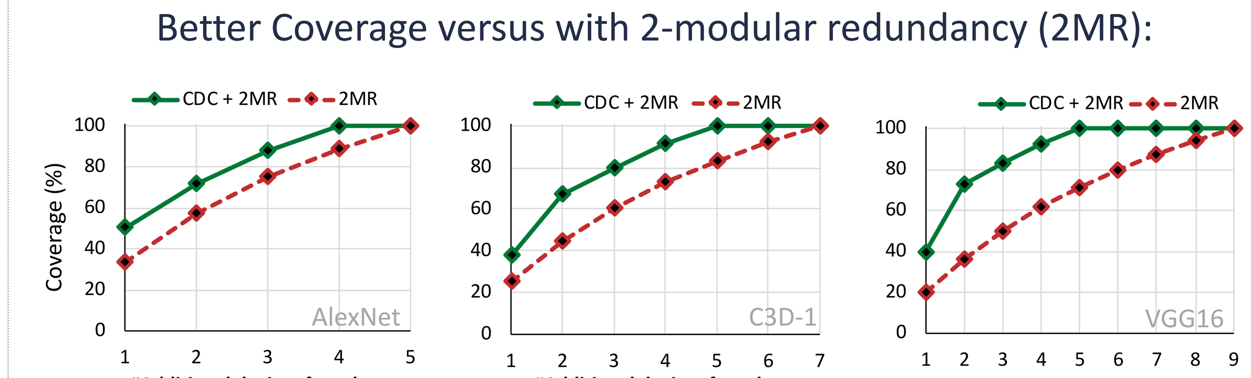
$$\begin{bmatrix} w_{11}+w_{(\frac{m}{2}+1)1} & w_{12}+w_{(\frac{m}{2}+1)2} & \cdots & w_{1k}+w_{(\frac{m}{2}+1)k} \\ w_{21}+w_{(\frac{m}{2}+2)1} & w_{22}+w_{(\frac{m}{2}+2)2} & \cdots & w_{2k}+w_{(\frac{m}{2}+2)k} \\ \vdots & \vdots & \ddots & \vdots \\ w_{\frac{m}{2}1}+w_{m1} & w_{\frac{m}{2}2}+w_{m2} & \cdots & w_{\frac{m}{2}k}+w_{mk} \end{bmatrix}_{\frac{m}{2}\times k}$$

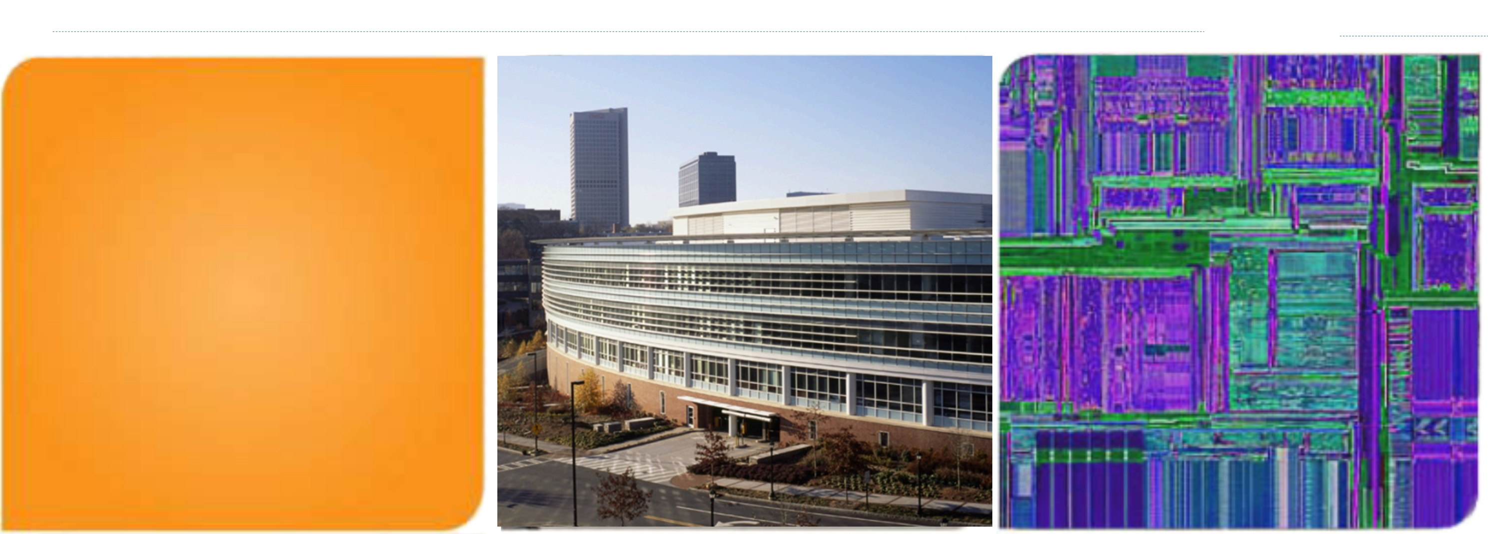
Straggler Mitigation & Failure Coverage

Do not need to wait for all devices to send data: (AlexNet)



Better Coverage versus with 2-modular redundancy (2MR):





NSF CSR 1815047

Georgia Tech comparch