



... for a brighter future



U.S. Department
of Energy

UChicago ►
Argonne_{LLC}

A U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC

TRANSIMS Training Course at TRACC

Transportation Research and Analysis Computing Center

Part 14

Parallelization and Partitioning

Dr.-Ing. Hubert Ley

Transportation Research and Analysis Computing Center

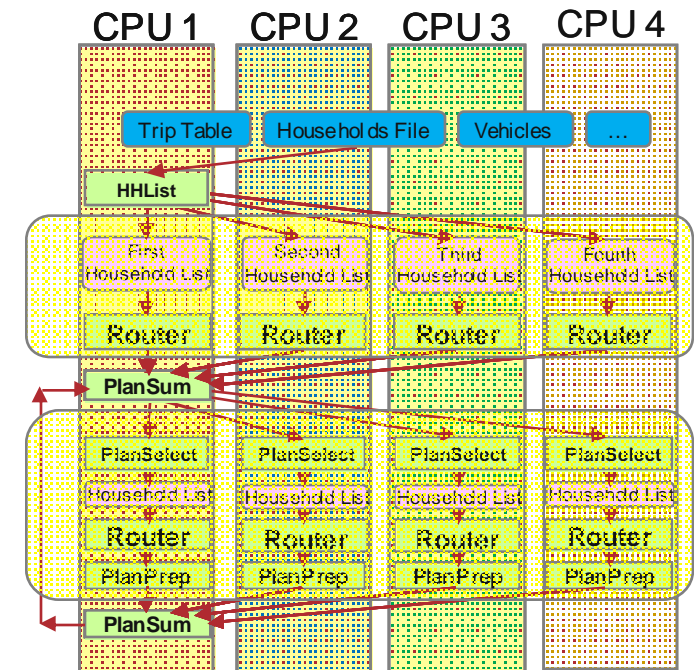
Last Updated: April 21, 2008

Contents

- Introduction to Partitioning
- Partitioned Model Step by Step
- The HHLIST Utility
- Tools that can be used in partitioned mode
- Complexity of Partitioning
- Coordination of Multiple Instances
- EXIT Return Codes
- Coordination of Execution Between Machines
- Partition-Aware Tools and their Control Files
- Control File Example: Router
- Control File Example: PlanSum
- Additional Comments on Partitioning

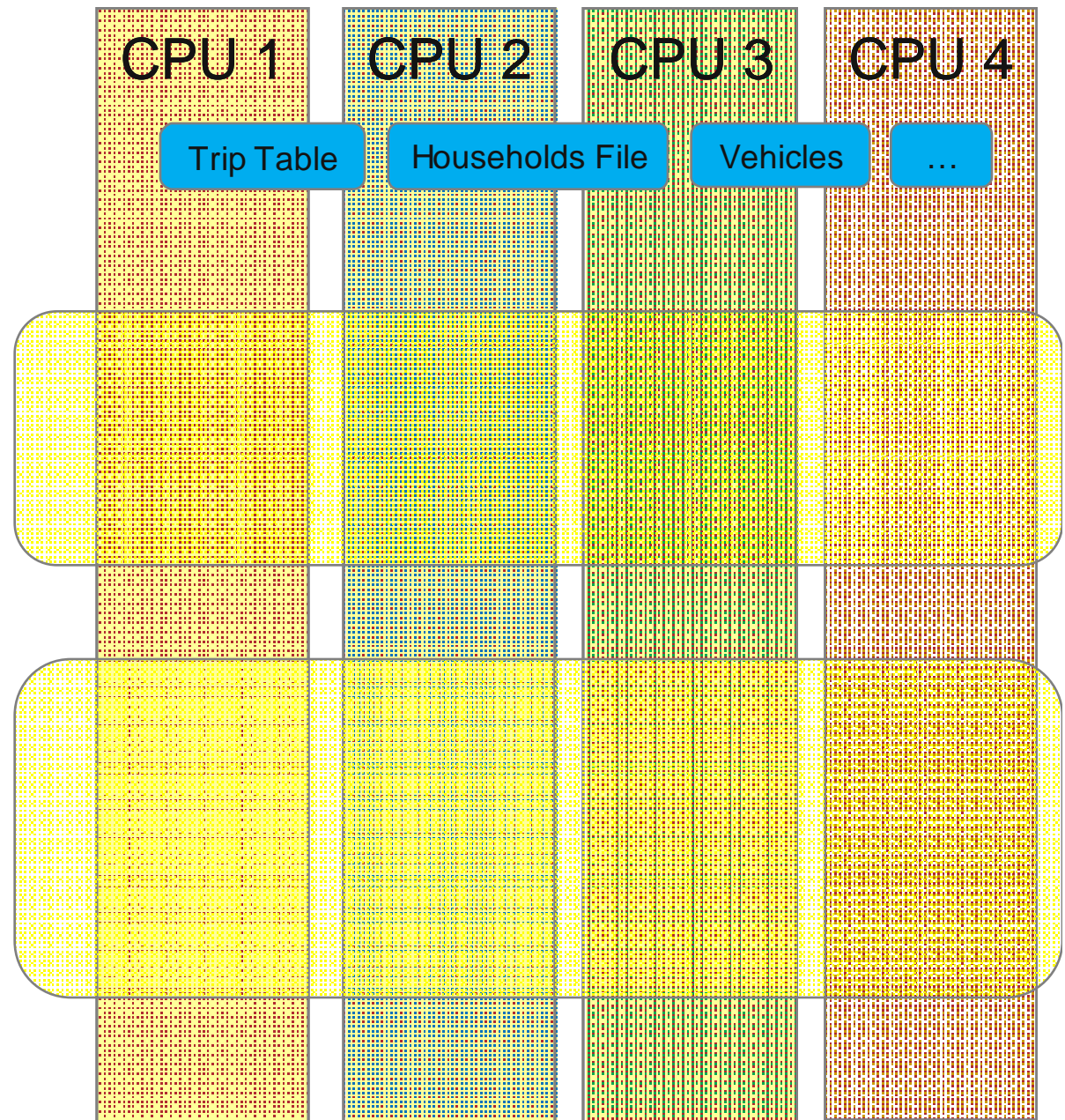
Partitioning

- Partitioning is based on the Router's capability of processing a subset of trips based on household lists
- Before routing, the special utility HHLList can be used to create random household subsets into simple household lists files
- Utilities such as PlanPrep, PlanSelect, and so on can simply work on the smaller subsets of data files
- PlanSum is a tool that usually runs on the entire dataset to calculate link delays based on BPR+ equations.
 - PlanSum runs in this scheme on a single CPU summarizing the results from all plans that the router calculated
 - The resulting link delay file is used as the basis for new routing in each partition



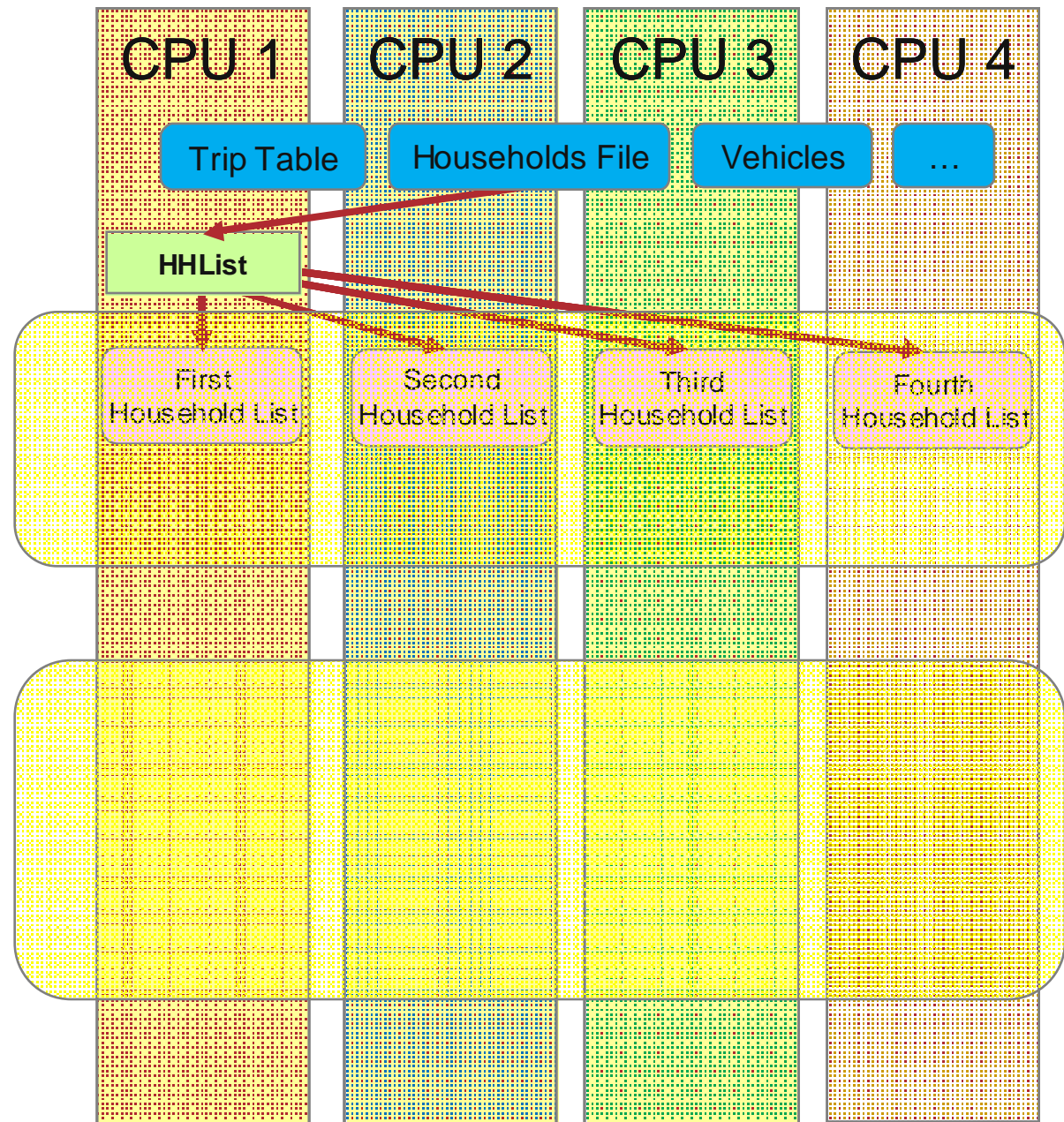
Partitioning

- In this example, we start with existing trip files or activity files
- Other files that are necessary to start routing are the
 - Household file
 - Vehicle file
 - Population file
 - Vehicle Type File
- These files are placed into a shared directory that is equally available to all CPUs, whether they are located on a single machine or several machines



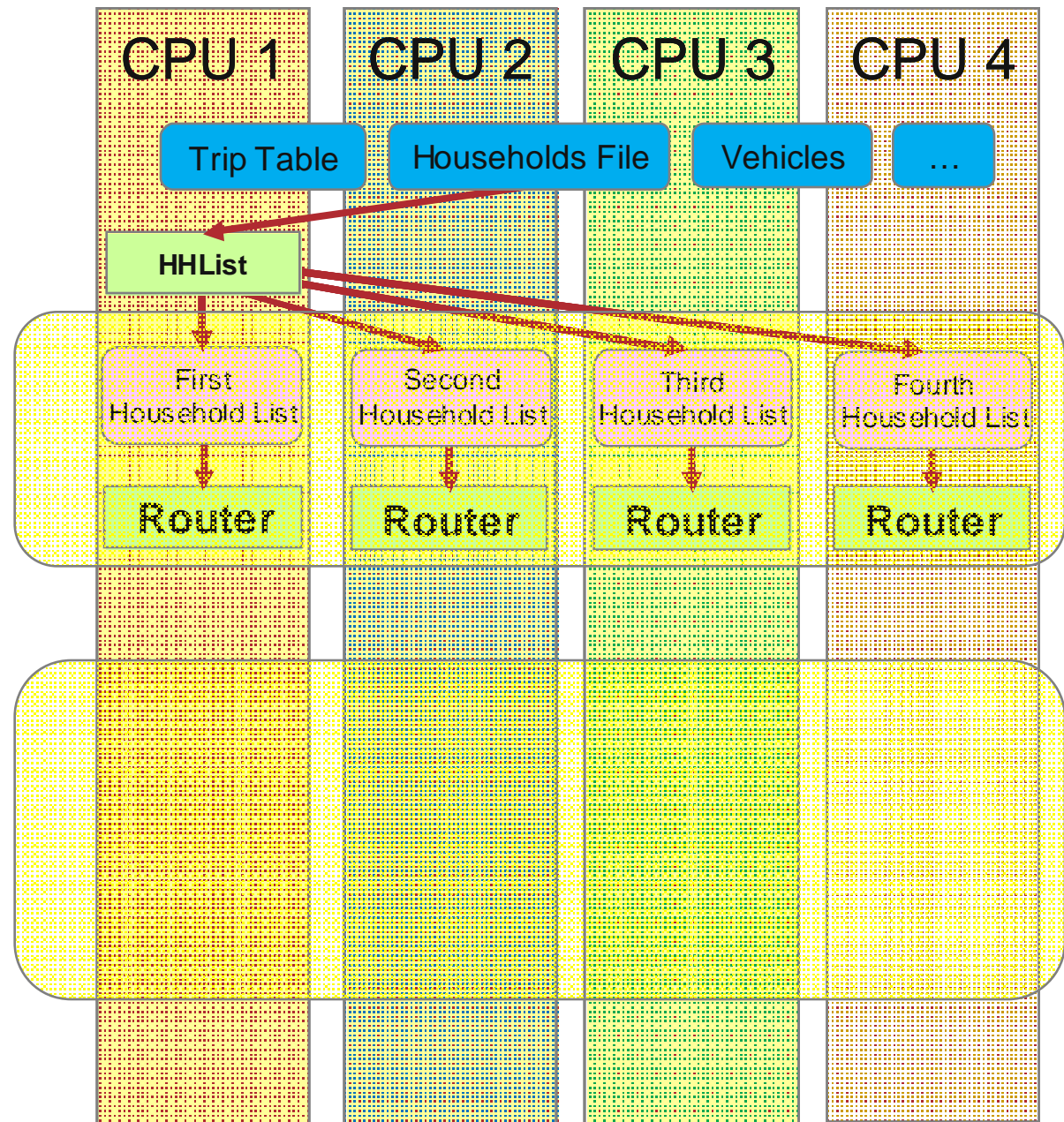
Partitioning

- The first step is to run an initial router iteration for the entire population as a starting point
- The plan is to run the router in 4 instances, each on a different subset of trips or activities
- Trips or activities are grouped by household as the smallest inseparable unit
- HHList can create a set of 4 household lists populated randomly from the existing households



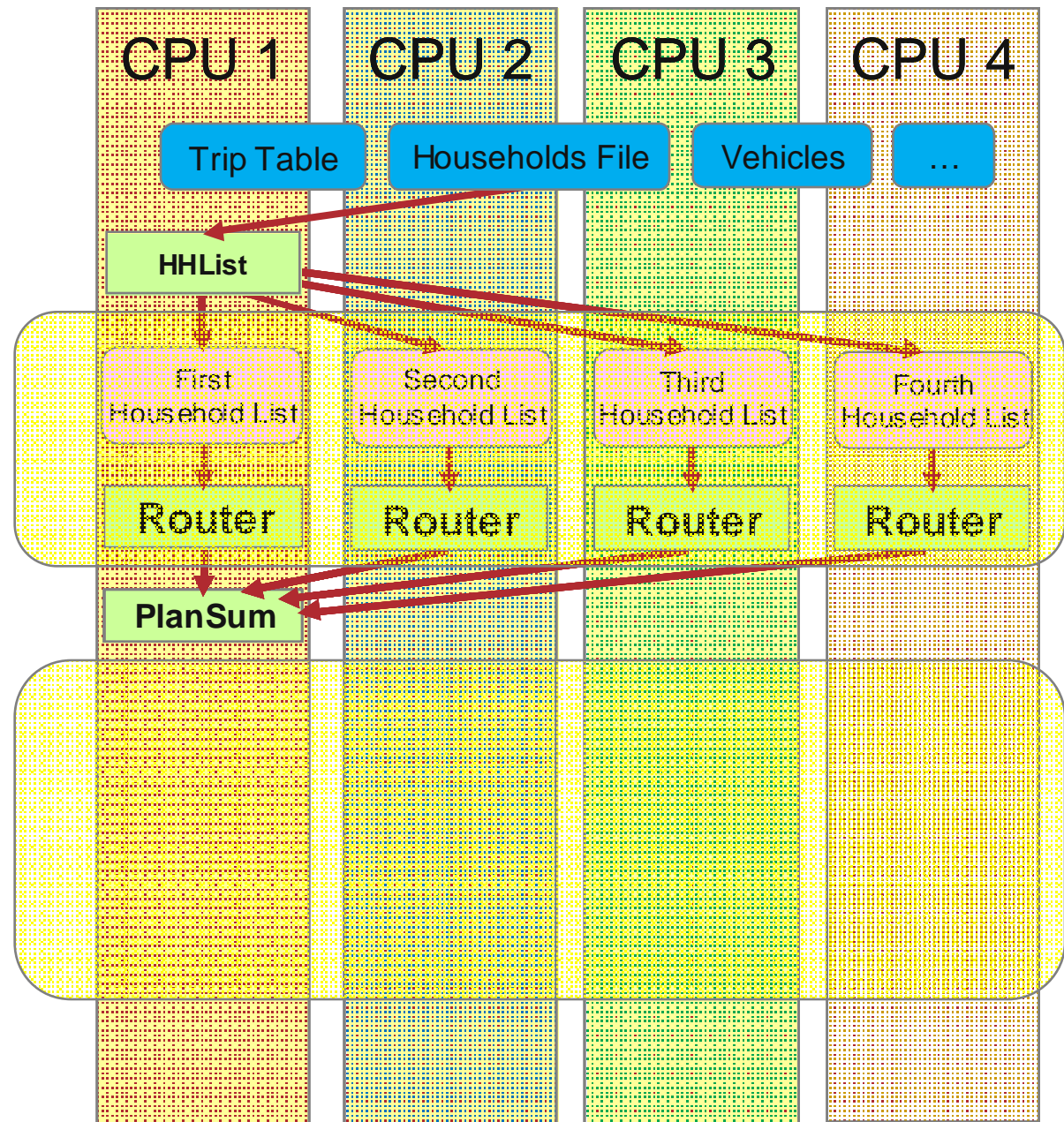
Partitioning

- Each instance of the router is started simultaneously on each CPU
- Each instance of the router reads the entire normal set of input files (shown in blue), and reads actually all records, so each router is doing exactly the same thing
- In addition, each router instance ignores all trips or activities that do not belong to its assigned list of households and writes a separate plan file for its own plans and problems



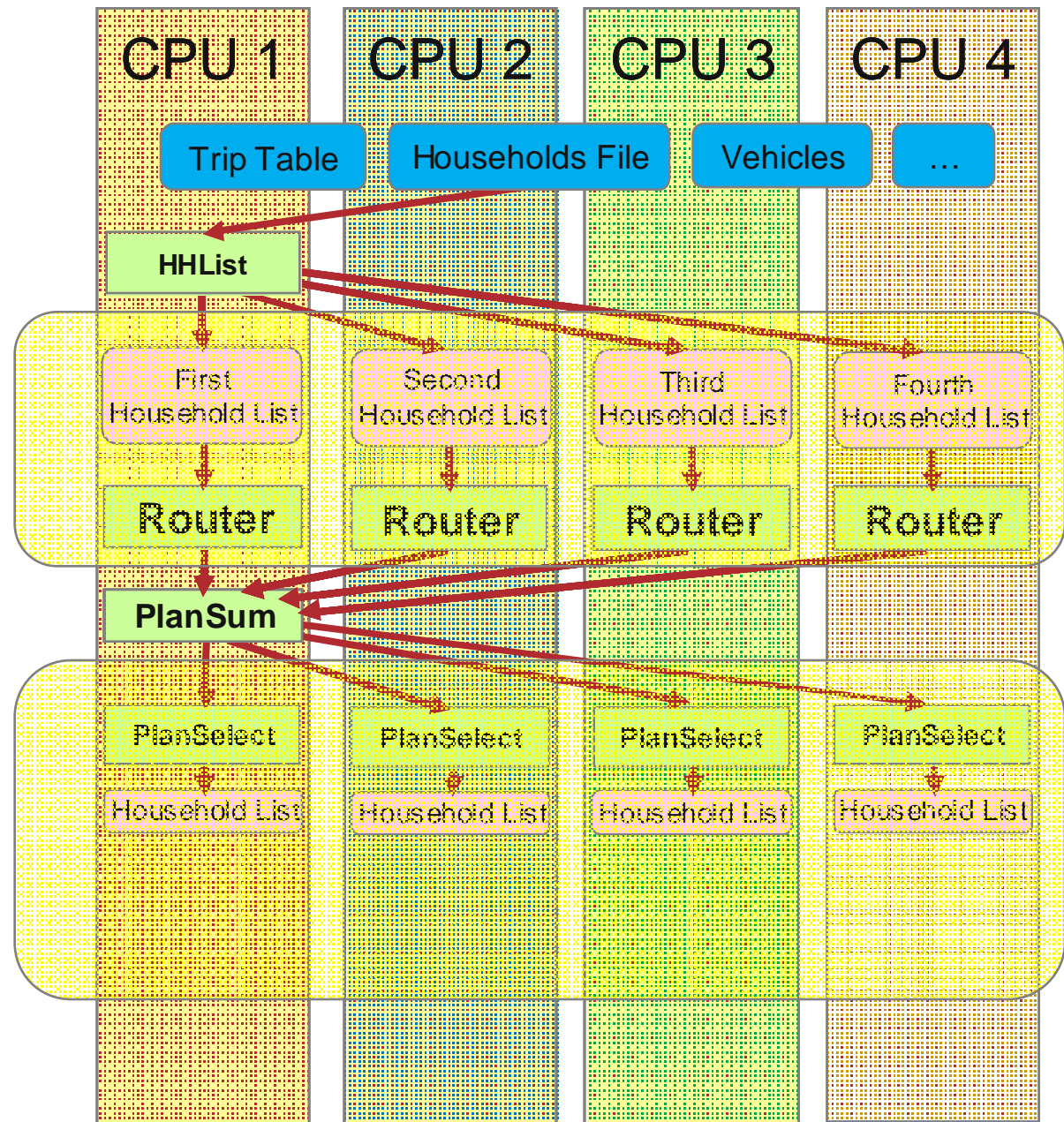
Partitioning

- Internal link delay updates are being used for each router instance with a link volume factor that corresponds to the number of processors
- The result of the first iteration are quite reasonable assignments of trips, but PlanSum is being used to read through all plans to create a single new link delay file
- Due to the need to evaluate all plans, PlanSum has to run on a single CPU



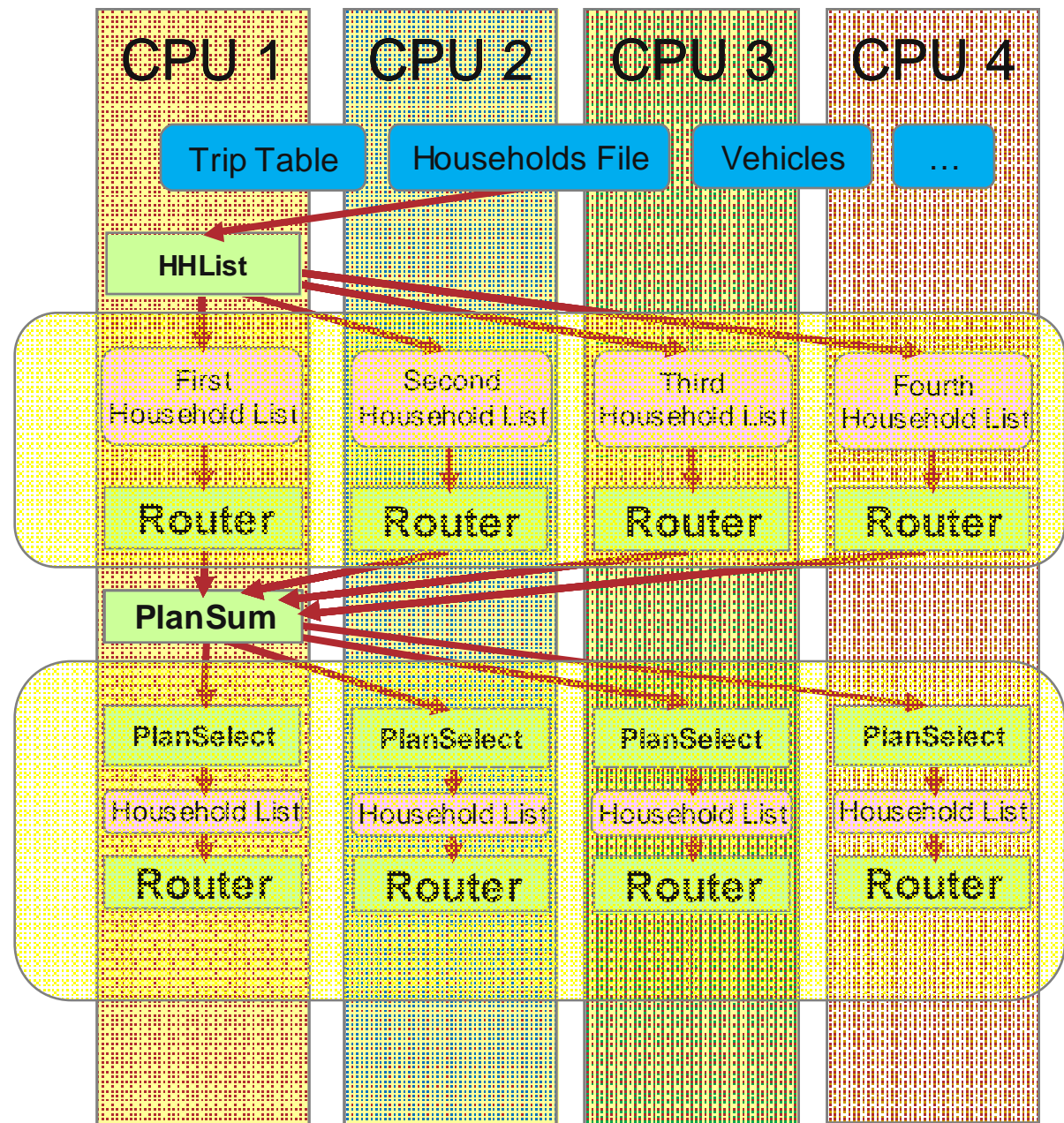
Partitioning

- To start additional iterations, PlanSelect is used to select eligible households for rerouting
- PlanSelect can apply its criteria just on the partitioned plan file created by the initial routing step
- The output from PlanSelect is a household list for each partition, which is simply a subset of the initial partition it is running in
- PlanSelect can obviously run in parallel for each partition



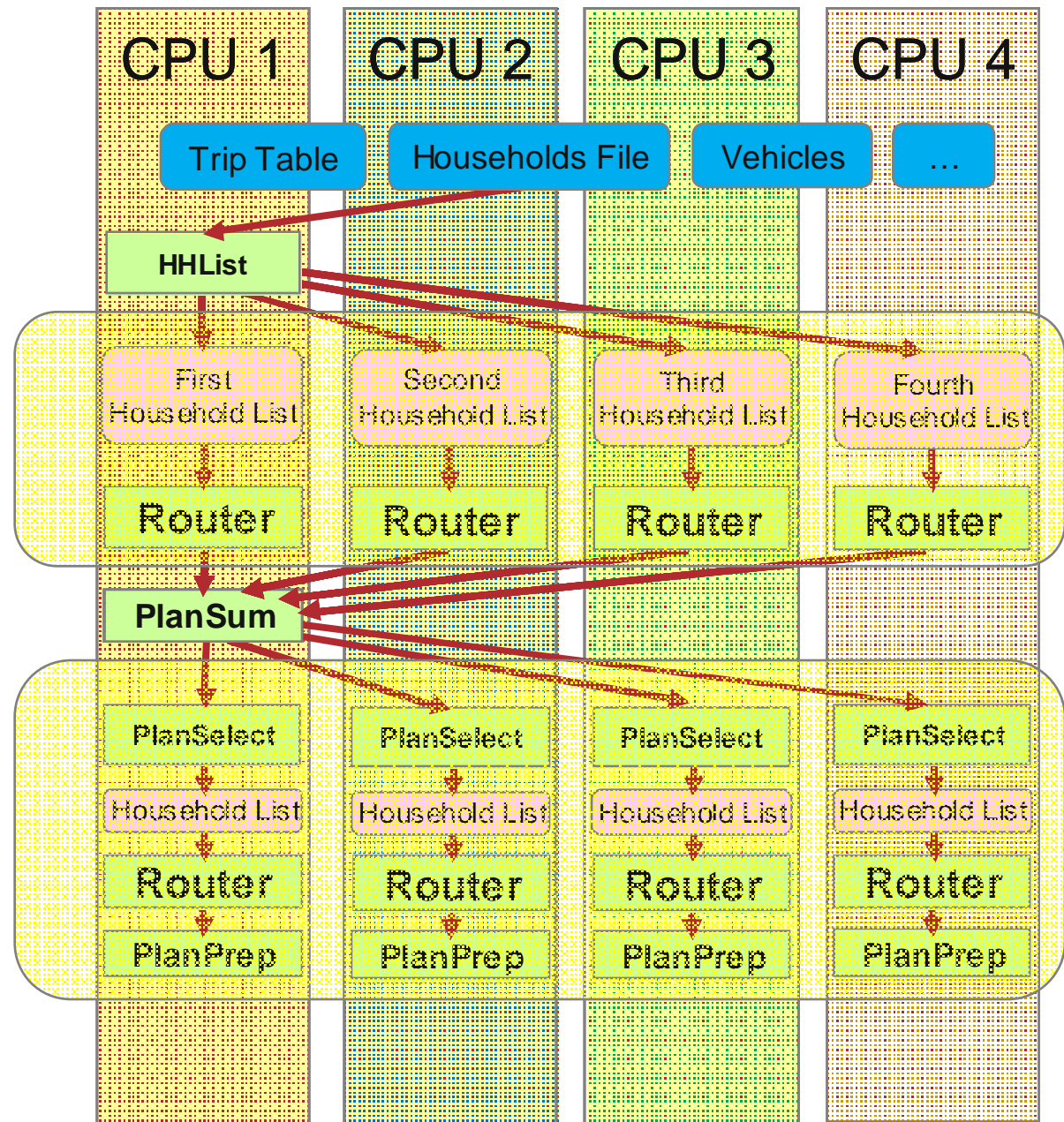
Partitioning

- The router is unaware of the “flavor” of household list it is given
- The household list is in this case just a subset of the household list defining the partition
- Just like the initial iteration, the router reads in all content from all input file and ignores trips and activities based on the household list
- Basis are not BPR+ equations but link delays
- The output is a plan file containing a subset of routes



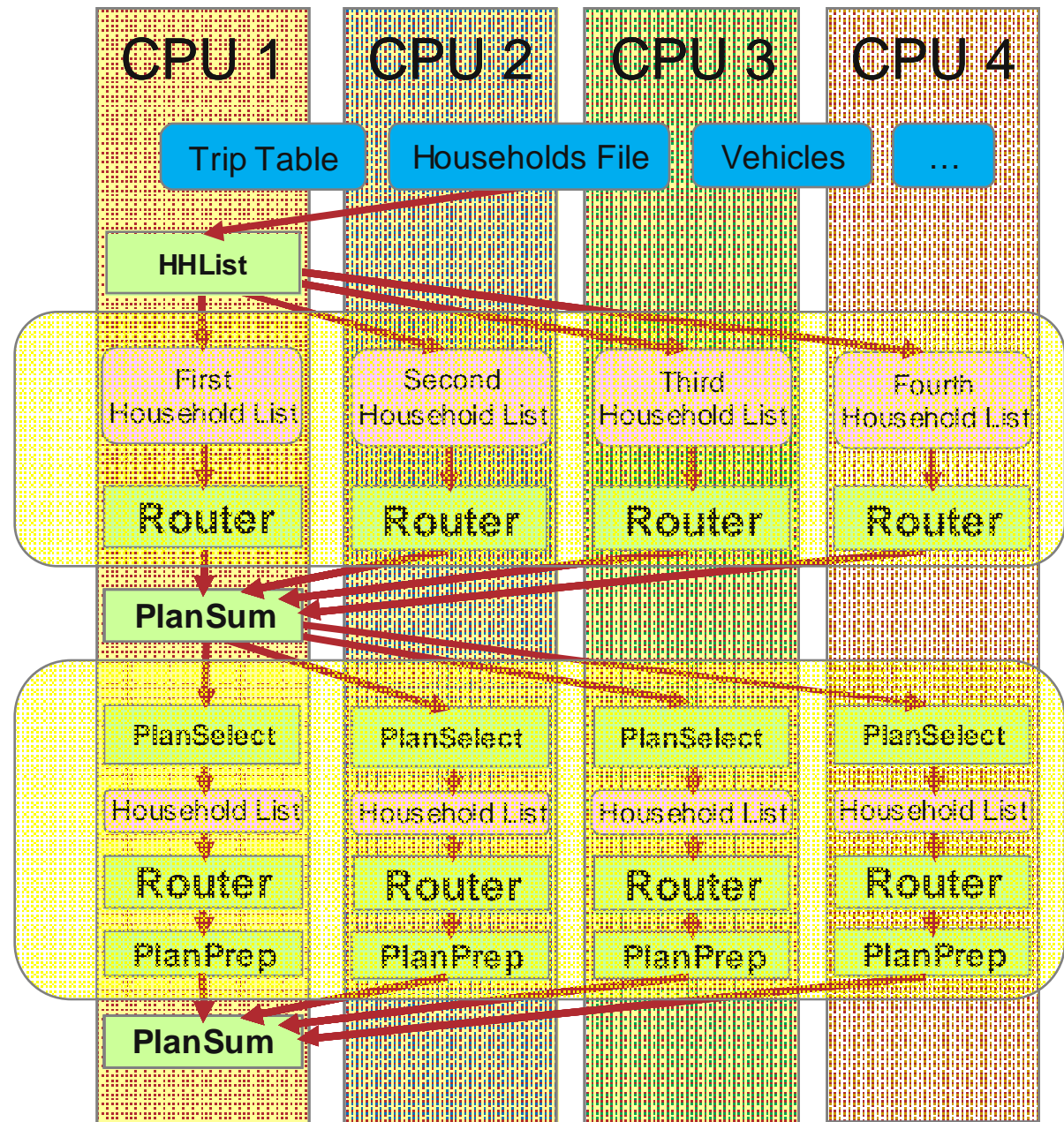
Partitioning

- The plans from the 4 partitions are covering just the subset of trips or activities selected for rerouting
- These new plans are merged into each partitions master plan file, replacing the previous records
- The result is a set of plan files that are complete, but up to date for the current iteration



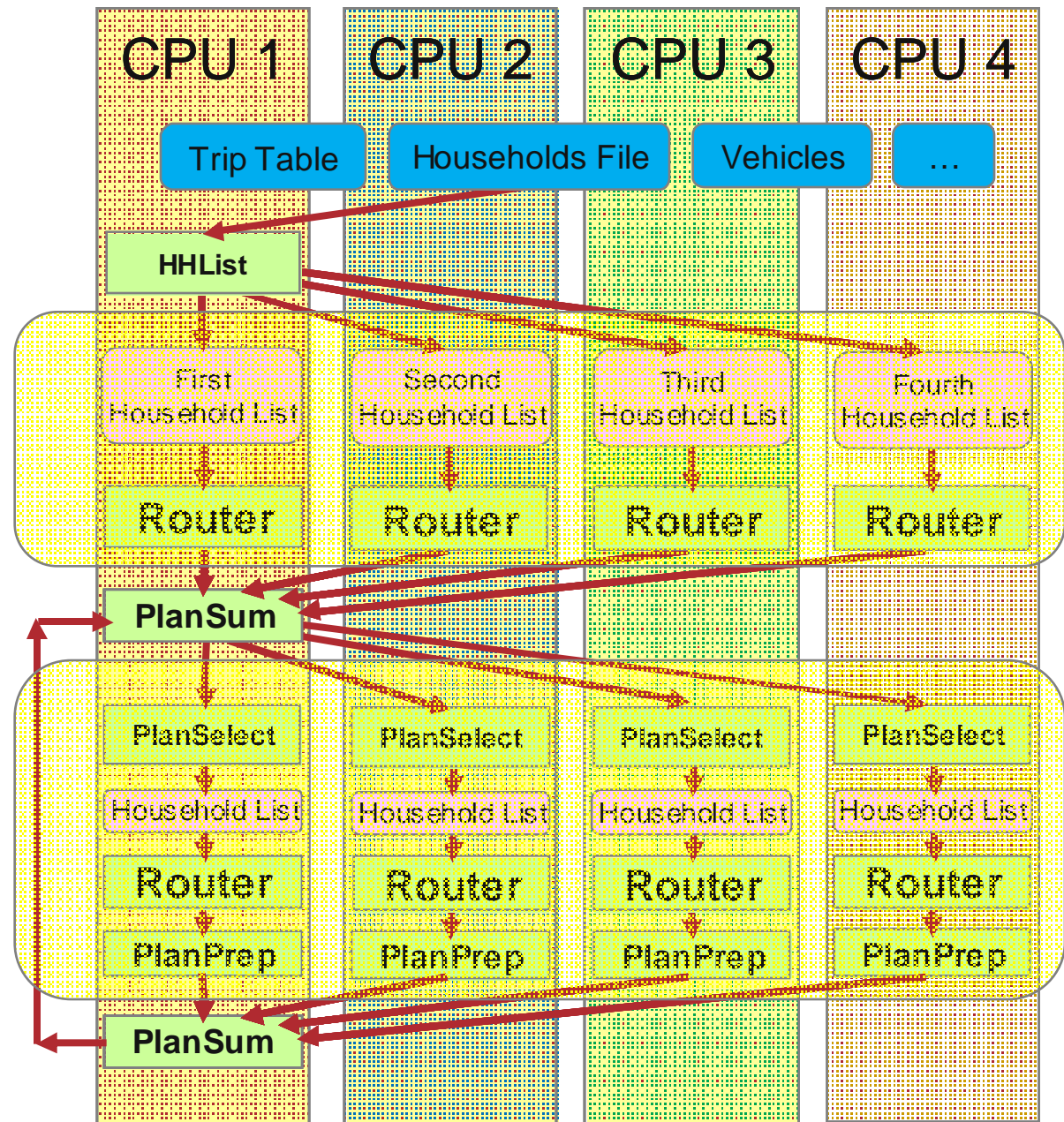
Partitioning

- The plans are now to be used to update the link delay file based on BPR+ equations
- This can only be done by analyzing all plans on a single CPU running PlanSum
- This run of PlanSum is technically identical with the previous run



Partitioning

- Therefore, we have reached the end of the iteration loop and can jump back to another parallel PlanSelect, Router, and PlanPrep iteration
- To run through the initial process for 28 million trips in the Chicago model, a single CPU requires 12.5 hours
- Running the initial router on 64 CPUs takes about 12 minutes
- Performance gain is close to linear



The HHList Utility

- HHList is a very simple program
 - It takes as input either a household list or a household file
 - It produces a number of household list files
 - The household numbers can be randomized
 - *This is important for load balancing, because ConvertTrips or the ActGen may have produced trip or activity tables that group certain kinds of trips together, some of which may have dramatically different characteristic*
 - No change to the Router is needed because it knows how to work on household lists anyway (e.g. for selective rerouting)
 - Other tools may be largely blind to partitioning because they may simply see data files that are subsets of the entire data
 - HHList itself is not a partitionable utility because it cannot work in parallel on multiple CPUs

Tools that can be used in partitioned mode

■ **Partitionable:**

- **ActGen**
- **AdjustPlans**
- **ArcPlan**
- **EventSum**
- **PlanCompare**
- **PlanPrep**
- **PlanSelect**
- **PlanSum**
- **PlanTrips**
- **PopSyn**
- **ProblemSelect**
- **Router**
- **SmoothPlans**
- **SubareaPlans**

■ **Not Partitionable:**

- | | | |
|------------------------|-------------------------|-----------------------|
| ■ <i>ArcDelay</i> | ■ <i>HHList</i> | ■ <i>Reschedule</i> |
| ■ <i>ArcDiff</i> | ■ <i>IntControl</i> | ■ <i>SideFriction</i> |
| ■ <i>ArcNet</i> | ■ <i>LineSum</i> | ■ <i>SmoothData</i> |
| ■ <i>ArcProblem</i> | ■ <i>LinkDelay</i> | ■ <i>SubareaNet</i> |
| ■ <i>ArcSnapshot</i> | ■ <i>LinkSum</i> | ■ <i>TPPlusNet</i> |
| ■ <i>ConvertTrips</i> | ■ <i>ListID</i> | ■ <i>TPPlusRoute</i> |
| ■ <i>CoordMatch</i> | ■ <i>LocationData</i> | ■ <i>TransimsNet</i> |
| ■ <i>DiurnalMatrix</i> | ■ <i>Microsimulator</i> | ■ <i>TransitNet</i> |
| ■ <i>EnhanceNet</i> | ■ <i>NewFormat</i> | ■ <i>TripSum</i> |
| ■ <i>ExportTransit</i> | ■ <i>ProblemSum</i> | ■ <i>Validate</i> |
| ■ <i>FileFormat</i> | ■ <i>Progression</i> | ■ <i>VehGen</i> |
| ■ <i>GISNet</i> | | |

Complexity of Partitioning

- Processing the data on multiple machines raises the problem of significantly increased complexity
 - Synchronization of all tasks is important
 - *What to do if the calculations fail for one partition?*
 - *How to make processors wait for the other partitions to finish?*
 - *Where are the log and output files are written?*
 - *How to combine some of the output files back into single entities?*
 - *How are the single entities such as the plan files are accessed by the processors?*
 - *Are there file sharing and access problems? File size limitations?*
 - Resource allocation issues
 - *Multiprocessor and Multi-Node environments (clusters) are usually a shared resource – how do you best access them?*
 - *How do you best deal with job queuing systems in shared environments?*

Coordination of Multiple Instances

- The coordination and control of multiple instances that have to function in a synchronized fashion needs to be carefully designed
- Simple monitoring in a single terminal window is not feasible
- Exit codes are explicitly returned by each individual executable to help with writing appropriate scripts
 - 0: **SUCCESS**
 - 1: **ERROR**
 - 2: **WARNING**
- Command line options are implemented to turn off potential interactive feedback
 - “-H” explains the options available in the executables
 - “-Q” executes the tools with minimal screen messages no interactive feedback
 - “-B” suppresses interactive feedback but writes progress and other screen messages
 - “-K” checks whether all control keys are actually supported

EXIT Return Codes

- In CMD batch scripts, the exit code is returned in the “%**ERRORLEVEL**” variable
 - The exact capability and mechanism of dealing with exit codes varies widely with every version of DOS and Windows – this works for XP!
 - *call ..\SetBinDir.bat*
 - *%BINDIR%\Router.exe -B -K InitialRouter.ctf 0*
 - *if %ERRORLEVEL% == 1 exit /b %ERRORLEVEL%*
 - *exit /b 0*
- Under Linux, the return codes are typically reported in the “**\$status**” variable
 - This may also depend on the particular shell that you are using for scripting (example is csh)
 - *#!/bin/csh*
 - *source ../SetBinDir.csh*
 - *\$BINDIR/Router -K -B InitialRouter.ctf 0*
 - *if (\$status == 1) exit(\$status)*

Coordination of Execution Between Machines

- This depends heavily on the hardware and software configuration
- All files should be available on high speed shared network file systems
- The file system must support concurrent reading without sharing violations
- There is no standard mechanism for process coordination
 - A file locking mechanism could be used (like PID files commonly used by daemons under Linux)
 - We are using a very simple MPI application at TRACC that coordinates executables on multiple threads and on multiple nodes equally
 - *The MPI process is brought up on all participating CPUs by the batch queuing system (for example PBS)*
 - *A call to the operating system's "system" function is issued to start a specific partition for each MPI rank*
 - *A barrier checks that all partitions are finished and returns the maximum exit code from any of the partitions*

Partition-Aware Tools and their Control Files

- The syntax for partition-aware tools includes a simple integer at the end of the control list indicating the partition number for this process
 - ***Router.exe -B -K InitialRouter.ctf 0***
 - This invocation starts the first partition (0-based integer)
- Control files are interpreted differently for partition-aware tools
 - File specifications for partitionable input and output files are internally extended by appending the extension “.tAA” to the names specified in the control file
 - “.tAA” is appended for the first partition (partition 0)
 - “.tAB” is appended for the next partition (partition 1)
 - and so on ...
 - That means that the control files are identical for each partition and there is no need to write N control files for N partitions
- Non-partitionable file specifications can be manually specified in control files using the extended “.tAA” logic to concatenate files transparently

Control File Example: Router

```
■ # Initial Routing for Everybody
■ TITLE                CMA Model at ANL/TRACC
■ REPORT_FILE          ../../log/00-router-Router.log
■ #
■ TRIP_FILE            ../../activity/Trip
■ VEHICLE_FILE         ../../vehicle/Vehicle
■ #
■ # Output files to be generated
■ # =====
■ #
■ NEW_PROBLEM_FILE      ../../router/00_problems
■ NEW_PLAN_FILE         ../../router/00_plans
■ NEW_PLAN_FORMAT       BINARY
■ NODE_LIST_PATHS       FALSE
■ #
■ NET_DIRECTORY         ../../network/production
■ NET_NODE_TABLE        FullArea_Node
■ NET_LINK_TABLE        FullArea_Link
■ ...
```

- The report file will be opened as “**00-router-Router_0.log**” for the first partition (notice the “**_0**” to indicate the partition
 - *This is not yet in the 4.01 release but part of 4.02*
- Trip and vehicle files will be opened as specified
- The problem file will be opened as “**../../router/00_problems.tAA**”
- The plan file will be opened as “**../../router/00_plans.tAA**”
- The network files will be opened as specified
- The user has to understand the underlying partitioning logic and which files are being opened using this scheme

Control File Example: PlanSum

```

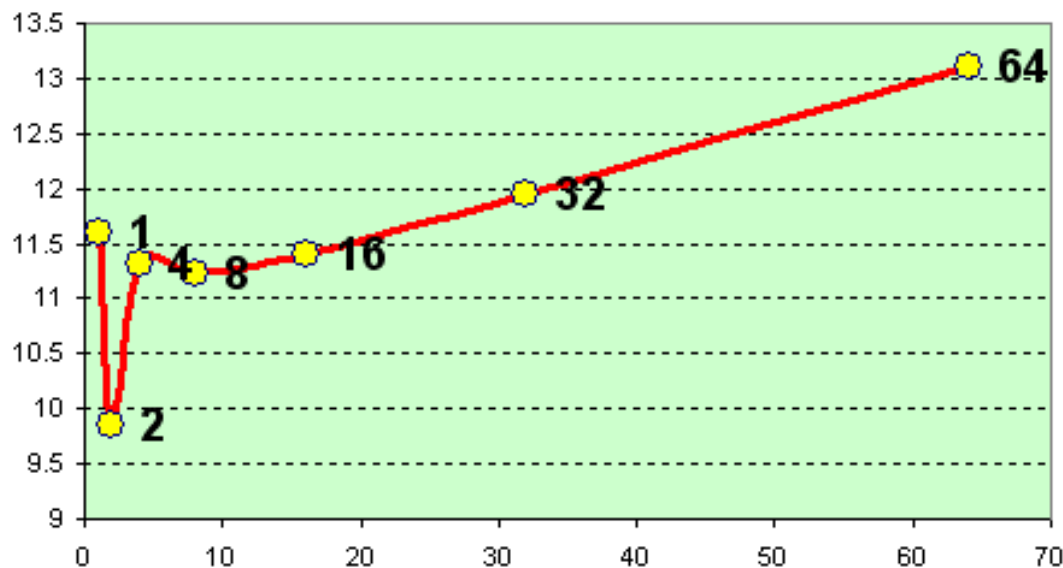
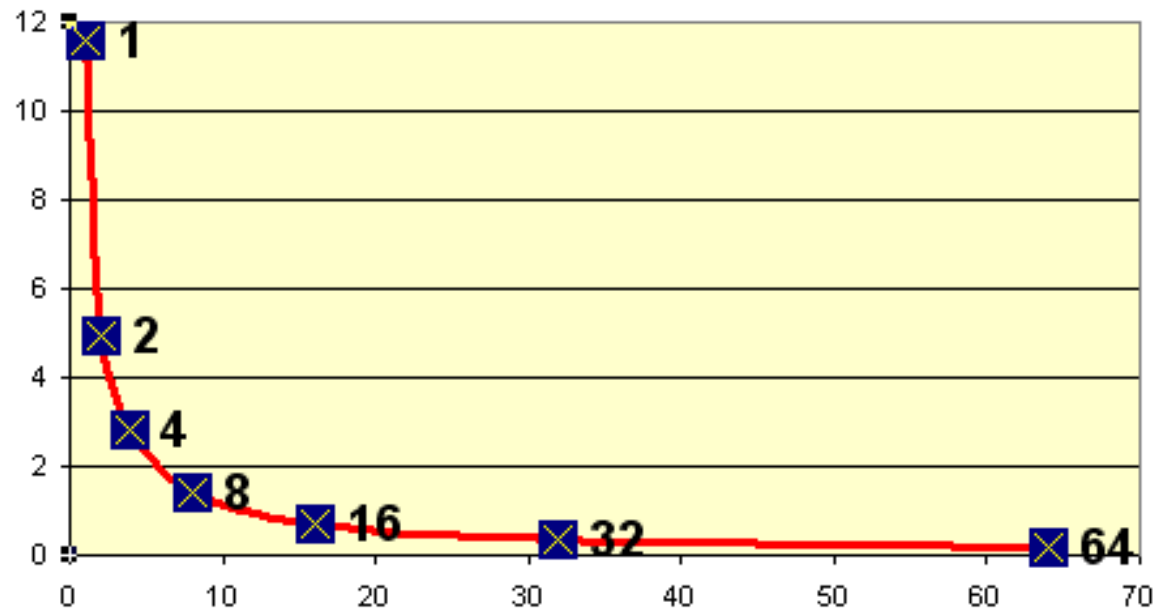
■ # Running PlanSum to create link delays
■ TITLE                CMA Model at ANL/TRACC
■ REPORT_FILE          ../../log/00-router-PlanSum.log
■ #
■ # Input files to read
■ # =====
■ #
■ PLAN_FILE             ../../router/00_plans.t*
■ PLAN_FORMAT          BINARY
■ NODE_LIST_PATHS      FALSE
■ #
■ # Input files to read
■ # =====
■ #
■ NEW_LINK_DELAY_FILE  ../../router/00_link_delay
■ NEW_LINK_DELAY_FORMAT BINARY
■ #
■ NET_DIRECTORY        ../../network/production
■ NET_NODE_TABLE       FullArea_Node
■ NET_LINK_TABLE       FullArea_Link
■ ...

```

- The report file will be opened as “**00-router-PlanSum.log**” because PlanSum is not running as a partitioned tool
- The plan file will be opened as “**../../router/00_problems.tAA**”, and upon the end of that file, “**../../router/00_problems.tAB**” is opened, and so on until the last file is encountered (all files are effectively concatenated)
- The link delay file will be opened as specified
- The network files will be opened as specified
- The user has to understand the underlying partitioning logic and which files are being opened using this scheme

Performance

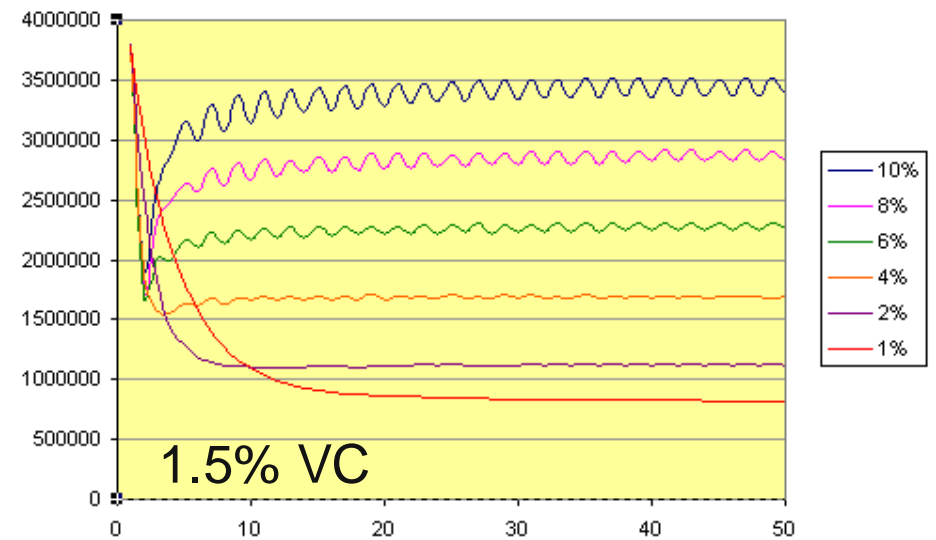
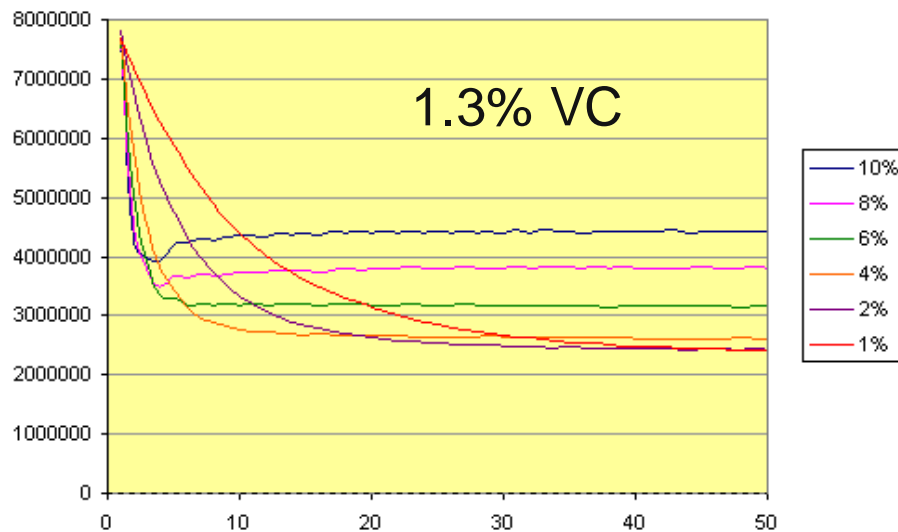
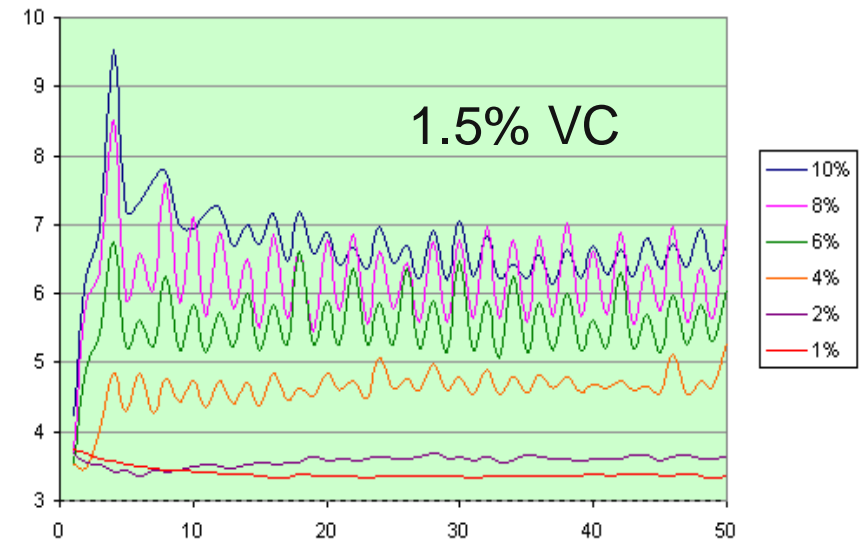
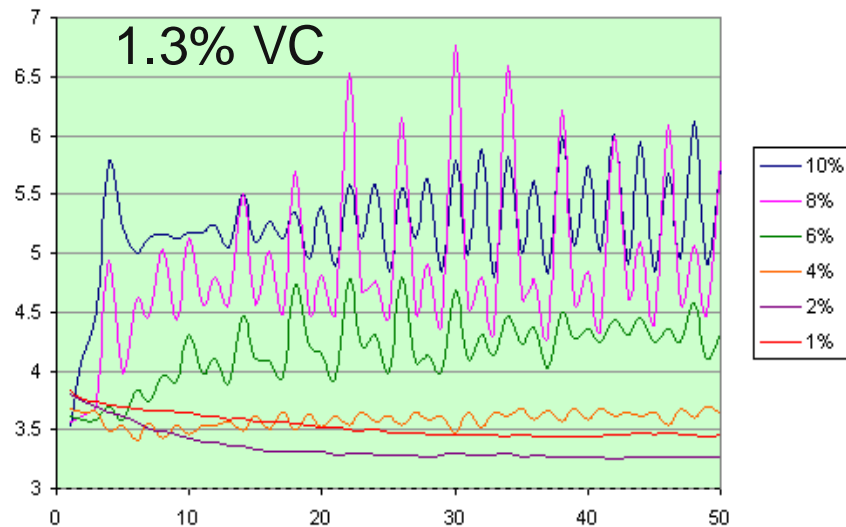
- The Chicago Model has been run on between 1 and 64 processors
- On a single CPU, it takes about 11.75 hours for the initial router run



- This is a dual core dual CPU environment, and all CPUs and cores have been utilized
- Utilizing additional CPUs leads a a fairly linear gain in performance

Parameter Studies

■ 50 Router Iterations Performance Study



Additional Comments on Partitioning

- Partitioning allows for great speed-ups at the cost of a more complex model
- Partitioned files and the option of concatenating them on input can also be used to write files under a certain size to accommodate operating system and file system limitations
 - FAT32 can only accommodate <4GB files
- Further documentation is necessary to fully understand the impact of the use of partitions across the TRANSIMS toolbox
- Unfortunately, the control file syntax does not indicate partitions, making it more difficult to understand what actions the tools will take in partitioned mode
- And last but not least, some tools create output files that are split in the same fashion as the input files are split, even if that is not necessarily desirable
- More work will hopefully streamline these powerful features

Credits and Acknowledgements

- GIS visualization materials were mostly developed at Argonne based on the TRANSIMS tools developed by AECOM for USDOT
- Chicago road and transit network data used in some of the examples was provided by the Chicago Metropolitan Agency for Planning
- USDOT provided the funding for the development of these training materials
- USDOT provided the funding for the TRACC computing center and the resources necessary to perform these training session