PARALLEL PROGRAMMING

TOPICS:

01. INTRO

MOTIVATION TIME, MONEY, COMPLEX PROBLEMS

- MOOR'S LAW LIMITS POWER CONSUMPTION

PARALLELIZATION:

- **AUTOMATIC** - NOT FEASIBLE TOOLS CAN'T EXTRACT ALL PARALLELISM

  BY HAND TOOLS ONLY COMPILE

TYPES OF PARALLELISM

  FLYNN'S TAXONOMY: INSTRUCTION, DATA SISD, MISD, SIMD, MIMD

- **LEVEL**
  - BIT
  - INSTRUCTION
  - TASK PARALLEL TASK GRAPH, PIPELINE, COMMUNICATION

PARALLEL PROGRAMMING

- NEW PROGRAMMING LANGUAGES ☹ COMPILERS, NEW LANGUAGE
- EXTENSIONS ☺ EASY DESIGN, COMPILER ☹ ONLY SOME TYPE OF PARALLELISM

02. PARALLEL PATTERNS 1

DEPENDENCIES EXECUTION ASSUMPTION, SEQUENTIAL CONSISTENCY

- **TRUE (FLOW) DEPENDENCE** RAW $out(S_1)$ Ç $in(S_2)\,^{1}$ Æ $S_1\ d\ S_2$
- **ANTI-DEPENDENCE** WAR $in(S_1)$ Ç $out(S_2)\,^{1}$ Æ $S_1\ d^{-1}\ S_2$
- **OUTPUT-DEPENDENCE** WAW $out(S_1)$ Ç $out(S_2)\,^{1}$ Æ $S_1\ d^{0}\,S_2$
- **LOOP-CARRIED DEPENDENCE**
  - CAN PREVENT LOOP ITERATION PARALLELIZATION
  - LEXICALLY FORWARD / BACKWARD

PARALLEL PATTERN = RECURRING COMBINATION OF TASK THAT SOLVES A SPECIFIC PROBLEM IN PARALLEL ALGORITHM DESIGN

NESTING PATTERN → ALLOWS OTHER PATTERNS TO BE COMPOSED IN A HIERARCHY

- PATTERN DIAGRAM VISUALIZES PATTERN IDEA

  ↳ TASK BLOCK = LOCATION OF GENERAL CODE IN AN ALGORITHM CAN BE ANOTHER PATTERN

SERIAL CONTROL PATTERNS: SEQUENCE, SELECTION, ITERATION, RECURSION (+ NESTING FOR COMBINATION)

PARALLEL CONTROL PATTERN EXTEND SERIAL, RELAXED ASSUMPTIONS

- **FORK-JOIN** FLOW FORKS INTO MULTIPLE PARALLEL FLOWS, THAT REJOIN LATER (SYNC ≠ BARRIER)
- **MAP** ELEMENTAL FUNCTION PERFORMED OVER EVERY ELEMENT OF A COLLECTION
- **STENCIL** ELEMENTAL FUNCTION ACCESSES SET OF NEIGHBORS – GENERALIZATION OF MAP
- **REDUCTION** COMBINER FUNCTION (ASSOCIATIVE) COMBINES EVERY ELEMENT IN A COLLECTION
- **SCAN** COMPUTES ALL PARTIAL REDUCTIONS OF A COLLECTION
- **RECURRENCE** COMPLEX VERSION OF MAP (LOOP ITERATION CAN HAVE DEPENDENCIES)

SERIAL DATA MANAGEMENT PATTERNS

- RANDOM READ/WRITE POINTERS REFER TO MEMORY ADDRESSES. ALIASING CAN CAUSE PROBLEMS
- STACK ALLOCATION FOR DYNAMIC ALLOCATION, ARBITRARY AMOUNT IN CONSTANT TIME. PRESERVES LOCALITY ☺
- HEAP ALLOCATION WHEN DATA CAN'T BE ALLOCATED IN LIFO. SLOWER, MORE COMPLEX ☹
- OBJECTS LANGUAGE CONSTRUCTS: DATA + CODE TO MANAGE THEM. PART OF CLASS OBJECTS

PARALLEL DATA MANAGEMENT PATTERNS

- PACK ELIMINATE UNUSED SPACE IN A COLLECTION + UNPACK
- PIPELINE PRODUCER-CONSUMER TASK CONNECTION
- GEOMETRIC DECOMPOSITION ARRANGE DATA INTO SUBCOLLECTIONS (DON'T MOVE DATA)
- GATHER READ COLLECTION OF DATA, GIVEN COLLECTION OF INDICES
- SCATTER WRITE OUTPUT AT GIVEN INDEX. POSSIBLE RACE CONDITIONS!
- OTHERS: SUPERSCALAR SEQUENCES, FUTURES, SPECULATIVE SELECTION, WORKPILE SEARCH, SEGMENTATION, EXPAND, CATEGORY REDUCTION

MAP

"FOR EACH" LOOP, EACH ITERATION INDEPENDENT ⇝ CAN BE APPLIED WITHOUT KNOWLEDGE OF NEIGHBORS

- SERIAL vs PARALLEL
- INDEPENDENCE + NO SHARED STATE OTHERWISE · RACES, UNDEFINED BEHAVIOR ☹
- OPTIMIZATIONS
  - SEQUENCES OF MAPS NO WRITE ALL INTERMEDIATE STAGES TO MEM!
  - CODE FUSION ☹ ARITHMETIC INTENSITY ☺ MEMORY USAGE
  - CACHE FUSION BREAK WORK INTO BLOCKS
- STENCIL MAP + NEIGHBORS
- WORKPILE WORK ITEMS ADDED TO MAP WHILE IN PROGRESS
- DIVIDE AND CONQUER BASE CASE SOLVED SERIALLY
- EXAMPLE: SCALED VECTOR ADDITION (SAXPY) $y = ax + y$

03. PARALLEL PATTERNS 2

COLLECTIVES DEAL WITH A COLLECTION AS A WHOLE (REDUCE, SCAN, PARTITION, SCATTER, GATHER)

REDUCE COMBINE COLLECTION OF EL INTO ONE SUMMARY VALUE

- COMBINER FUNCTION ASSOCIATIVE, ELEMENTS PAIRWISE
- VECTORIZATION, TILING
- PRECISION WITH FLOAT ☹
- EXAMPLE: DOT PRODUCT MAP(*) + REDUCE(+)
- EXAMPLE: MERGE SORT MAP-REDUCE
    - RIGHT BIASED ☹ $O(m^2)$, INSERTION SORT
    - TREE-SHAPED SORT $O(m \log m)$ ☹ NOT A LOT TO PARALLELIZE

SCAN GENERATES NEW SEQUENCE WITH PARTIAL REDUCTIONS

- INCLUSIVE / EXCLUSIVE
- EXAMPLE: RADIX SORT, QUICKSORT
- PARALLELIZATION: UP SWEEP + DOWN SWEEP, 3 PHASE SCAN WITH TILING

04. PARALLEL PATTERNS 3

DATA MOVEMENT LIMITS PERFORMANCE MORE THAN COMPUTATION

- ACROSS MEMORY LOCALITY, DATA ORGANIZATION
- ACROSS NETWORKS NUMBER OF MESSAGES

GATHER READ INPUT COLLECTION AT INPUT INDICES → WRITE OUTPUT COLLECTION

- "MAP + RANDOM READ"
- OUTPUT EL. TYPE = SOURCE EL. TYPE
- OUTPUT SHAPE = INDEX COLLECTION SHAPE

- SPECIAL CASES
  - SHIFT + DEFAULT VALUE, DUPLICATE, ROTATE
  - ZIP example: REAL + IMAGINARY PARTS → COMPLEX NUMBERS ⇝ COMBINES IN PAIRS
  - UNZIP

SCATTER

- "MAP + RANDOM WRITES"
- COLLISIONS PARALLEL WRITES TO THE SAME LOCATION
  ↳ UNDEFINED RESULT => NEED RULES
- **COLLISION RESOLUTION**
  - ATOMIC SCATTER NON-DETERMINISTIC
  - PERMUTATION SCATTER ILLEGAL COLLISIONS -> OUTPUT PERMUTATION OF INPUT (=> GATHER)
  - MERGE SCATTER ASSOCIATIVE AND COMMUTATIVE OPERATIONS TO MERGE
  - PRIORITY SCATTER PRIORITY BASED ON ITS POSITION

PACK TO ELIMINATE UNUSED ELEMENTS FROM MEM + MOVED TO CONTIGUOUS MEM

- BOOL ARRAY, ~CUMSUM, WRITE OUTPUT WITH OFFSET!

UNPACK

GENERALIZATION OF PACK

- SPLIT ELEMENTS MOVED LEFT/RIGHT, NO INFO LOST
  UNSPLIT
- BIN SPLIT WITH MORE CATEGORIES

MAP + PACK

- EXPAND EACH EL CAN PRODUCE ANY NUMBER OF EL + RES FUSED TOGETHER

PARALLELIZING ALGORITHMS DIVIDING DATA INTO SECTION

- DIVIDE AND CONQUER
- FORK - JOIN

- GEOMETRIC DECOMPOSITION
- PARTITIONING NON OVERLAPPING, EQUAL-SIZED REGIONS
- SEGMENTATION NON OVERLAPPING, NON-UNIFORM REGIONS

AoS ARRAY OF STRUCTURES
SoA STRUCTURE OF ARRAYS BETTER FOR VECTORIZATION

05. PTHREADS, OPENMP 1

1. DESIGN PARALLEL ALGORITHM UNDERSTAND DATA DEPENDENCIES
2. DESIGN PARALLEL PROGRAM ANALIZE TARGET ARCHITECTURE, CHOOSE LANGUAGE, ANALYZE COMMUNICATIONS

EVALUATION
- SEQUENTIAL ALGORITHMS → **TIME, MEMORY COMPLEXITY**
- PARALLEL ALGORITHMS → **TIME, RESOURCE COMPLEXITY**

QUALITATIVE EVALUATION: DAG, WORK, SPAN, PARALLELISM

PP MODELS SHARED MEMORY, THREADS, MESSAGE PASSING/DISTRIBUTED MEM, DATA PARALLEL
- VERILOG/VHDL ☺ COMMUNICATION+MEM CONTROL, NO OVERHEAD ☹ SPECIFIC HW, DIFFICULT TO LEARN, NO DIFF ARCH
- MPI ☺ DIFFERENT ARCH, SYNCH + DATA COMM EXPLICITLY MANAGED ☹ COMM OVERHEAD, MORE DIFFICULT
- PTHREAD ☺ DIFFERENT ARCH, EXPLICIT PARALLELISM, APP CONTROL ☹ TASK MANAGEMENT OH, NOT SCALABLE, LOW-LEVEL API
- OPENMP ☺ EASY LEARN, SCALABLE ☹ ONLY SH-MEM HOMOGENEOUS SYS, SMALL TASK INTERACTION
- CUDA ☺ GPU, EASY KERNELWRITING, OPTIMIZED LIBRARIES ☹ ONLY NVIDIA GPU, DIFFICULT MASSIVE PAR. + KERNEL OPT
- OPENCL ☺ TARGET INDEPENDENT, HIDES ARCH DETAILS ☹ DIFFICULT TO PROGRAM + OBTAIN BEST PERFORMANCE
- APACHE SPARK ☺ API FOR DIFFERENT LANG, NO EXPLICIT PARALL/COMM ☹ ONLY FOR BIG DATA APP, NO GPU FULL SUPPORT
- MIX OPENMP + CUDA, MPI + OPENMP, OPENCL + VERILOG/VHDL

PROCESS INFO ABOUT RESOURCES, EXEC STATE
THREAD INDEPENDENT STREAM OF INSTR WITHIN A PROCESS
- CAN EXECUTE AT THE SAME TIME

- HAS LOCAL RESOURCES, CAN ACCESS SHARED PROCESS RESOURCES
- IMPLICIT COMMUNICATION (SHARED VAR) -> EXPLICIT SYNCHRONIZATION TO AVOID **DATA RACE**

THREADED PROGRAM

- IF ORGANIZABLE IN DISCRETE CONCURRENT TASKS, THREAD SAFE
- MODELS: MANAGER/WORKERS, PIPELINE
- STANDARD IMPLEMENTATIONS: PTHREAD, OPENMP

PTHREADS

THREAD MANAGEMENT CREATE, JOIN/DETATCH, BARRIER INIT/WAIT, CANCEL/EXIT
- SYNCHRONIZATION MUTEX, CONDITION VARIABLES
- examples CALCULATION OF π (MONTECARLO), MULTIRATE BAND-PASS FILTER, PRODUCER-CONSUMER

OPENMP API FOR MULTI-THREADED SHARED-MEM PROGRAMMING

- BASED ON FORK-JOIN MASTER + CONCURRENT SLAVES
- LANG EXTENSIONS: PRAGMA PREPROCESSOR DIRECTIVES, IDENTIFY TASKS. IMPLIED BARRIER AT THE END
  - PARALLEL [NUM THREADS, IF]
  - WORK SHARING: FOR, SECTIONS, SINGLE/MASTER, TASK

06. OPENMP 2

OPENMP LANGUAGE EXTENSIONS

- SYNCHRONIZATION: CRITICAL [NAME], BARRIER, ATOMIC
- DATA ENVIRONMENT: DATA SCOPE ATTRIBUTE CLAUSES (PRIVATE, SHARED, DEFAULT, REDUCTION)
  - SEQUENTIAL MEM CONSISTENCY LOAD/STORE IN ORIGINAL SEQUENTIAL ORDER
  - RELAXED MEM CONSISTENCY EACH THREAD TMP VIEW. FLUSH -> TO HAVE SAME GLOBAL VIEW
- RUNTIME FUNCTIONS GET NUM THREADS, GET THREAD NUM, SET NUM THREADS, GET WTIME, GET WTICK
- examples: CALCULATION OF π (INTEGRAL)

NESTED PARALLELISM PARALLEL, WORKSHARING ⇝ BUT ID RESTARTS FROM 0!
THREAD CANCELLATION CANCEL + CANCELLATION POINT

07. OPENMP 3

OPENMP TASK = BLOCK OF CODE IN A PARALLEL REGION, THAT CAN BE EXECUTED SIMULTANEOUSLY WITH OTHER TASKS IN THE SAME REGION

- **PIPELINE**
  - SECTIONS ☹ COMMUNICATION AND LOAD IMBALANCE PROBLEM
  - TASKS ☺ CAN EXECUTE WHENEVER INPUT IS READY
- TASKWAIT / TASKGROUP
- DEPEND IN/OUT
  PRIORITY IMPORTANT TASKS TO BE EXECUTED MORE FREQUENTLY
- TASKLOOP

SIMD VECTORIZATION INSTRUCTIONS ON BLOCK OF DATA (VECTORS)

- COMPILER ISSUES LOOP DEPENDENCIES, POINTER ALIASES, DATA ALIGNMENT, LOOP BOUNDS
- VECTOR REGISTERS LOOP CHUNK SHOULD FIT IT
- simdlen PREFERRED SIZE (PERFORMANCE)

HETEROGENEOUS ARCHITECTURES

- #pragma omp target [map(...)]
  - CODE TO ACCELERATOR, IF PRESENT
  - HOST THREAD BLOCKED
  - map COPIES VARIABLES IN TARGET MEMORY (IT'S SHARED) + to, from, tofrom
- BATCH PROCESSING/COOPERATIVE PATTERNS

08. MPI 1

SHARED MEM ARCHITECTURES ☹ HARD TO BUILD ☺ EASY TO PROGRAM (OPENMP, C++ LIBRARY)

DISTRIBUTED MEM ARCHITECTURE REQUIRES EXPLICIT COMMUNICATION ☹ SYNCHRONIZATION

MPI (MESSAGE PASSING INTERFACE) STANDARD THAT DEFINES INTERFACES FOR DISTRIBUTED MEM COMMUNICATIONS

- OUTCOME: PDF DOCUMENT
- IMPLEMENTATIONS OPENMPI, MPICH
  - libmpi.so IMPLEMENTATION LIBRARY
  - mpicc COMPILER
  - mpiexec LAUNCHER
  - mpirun FORK + EXEC CODE
- INIT + THREAD USAGE
- FINALIZE
- COMMUNICATOR CONTEXT

POINT TO POINT COMMUNICATION

- SEND, RECV (STATUS), PROBE
- WAIT, TEST, CANCEL

09. MPI 2

COLLECTIVE COMMUNICATIONS

- COMMUNICATOR HANDLING DUP, SPLIT, FREE
- SYNCHRONIZATION BARRIER
- DATA TRANSFER BROADCAST, GATHER, SCATTER

  REDUCTIONS REDUCE (MAX, SUM, LAND, MAXLOC)

10 HALIDE

POWER = OP / SECONDS · JOULES / OP

GENERAL-PURPOSE PROCESSORS NOT ENERGY EFFICIENT

HETEROGENEOUS PROCESSING

- **GPU**
- **DSP** DIGITAL SIGNAL PROCESSORS
- **ASIC** DOMAIN-SPECIFIC HW EXTREME SPECIALIZATION
- **FPGA**

CHALLENGES — SW DESIGN ALGO THAT CAN BE DECOMPOSED AND MAP WELL ON HW

CHALLENGES — HW CHOOSE RESOURCES, CHIP AREA FOR EACH FUNCTIONALITY

ENERGY-EFFICIENT COMPUTING REDUCE COMPUTATION, DATA TRANSFER , CHOOSE BEST COMPUTE UNIT

DSL PRODUCTIVITY + PERFORMANCE, NO GENERALITY

IMAGE PROCESSING DEMANDS OPTIMIZATION → FASTER ALGO, HW, MORE EFFICIENT USE OF HW

- C++ WITH MULTITHREADING
- CUDA / OPEN CL
- OPTIMIZED LIBRARIES

HALIDE DSL EMBEDDED IN C++ + SPECIALIZED COMPILER

- DECOUPLE ALGO FROM SCHEDULE
- ☹ ALL COMPUTATIONS OVER REGULAR GRIDS, MANUALLY SPECIFY SCHEDULE (AUTOSCHEDULER CAN SUGGEST)
- CAN OPTIMIZE WITH
  - ORDER OF VALUES COMPUTED SERIAL, VECTORIZE, TILING
  - WHEN COMPUTE RESULTS eg: COMPUTE ALL/SUBSET OF PRODUCER, THEN PASS TO CONSUMER
- SCHEDULING ACROSS STAGES DRIVEN BY CONSUMER
- examples: BLUR, BRIGHTEN

11. PARALLEL PATTERNS APPLIED

PP MODELS OFFER API FOR PATTERNS

- MAP #pragma omp parallel for
- REDUCTION #pragma omp for reduction (+ : var)
- WORKPILE example : TREE SEARCH
- SCAN + 3 PHASE TILING APPROACH

+ SORTING