20BCE1025
Abhishek N N
abhishek.nn2020@vitstudent.ac.in

Lab2 - OpenMP Basics & Vector addition

i) OpenMP Case1 1000 threads by using runtime lib routine

Code:

```c
#include <omp.h>
#include <stdio.h>

int main(int argc, char *argcv[]) {
      int pid = 0, numthreads = 1000;
      omp_set_num_threads(numthreads);

      #pragma omp parallel private(pid)
      {
            printf("%d ", omp_get_thread_num());
            if (omp_get_thread_num() == pid) {
            printf("master");
      }
      }
      return 0;
}
```

Output:

```
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ gcc -fopenmp -o case1 case1.c
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ ./case1
13 57 295 751 693 394 695 305 20 753 395 696 309 93 756 21 397 699 311 96 895 393 398 313 336 98 391 317 399 335 104 390 325 400 334 1 3
89 109 333 401 332 3 388 110 357 402 331 4 387 365 114 403 2 386 330 367 116 404 5 384 329 373 405 117 328 6 382 375 406 327 120 380 7 3
77 407 326 379 124 8 381 408 324 378 9 383 126 410 323 376 10 385 411 322 127 374 11 392 412 321 128 372 14 396 413 320 371 129 15 580 4
14 319 370 585 131 415 17 318 369 588 416 133 368 316 18 591 417 315 366 134 19 596 418 314 364 24 144 599 419 312 363 22 601 145 420 31
0 362 23 605 421 308 146 361 26 615 422 307 360 147 27 623 424 306 359 28 149 629 426 304 358 30 633 150 303 356 427 29 639 355 428 154
302 31 641 354 429 33 301 156 353 645 430 32 300 159 352 649 431 34 299 142 653 160 432 35 298 661 161 433 162 36 959 164 664 435 961 16
5 37 167 708 964 437 38 168 737 968 170 438 39 171 971 694 439 172 41 173 440 973 742 176 40 441 975 178 42 180 745 442 977 43 181 748 1
82 443 980 44 750 183 185 444 982 45 752 445 119 984 46 754 158 101 446 700 647 825 652 197 646 821 701 827 654 644 194 819 668 829 655
200 643 818 199 656 831 642 816 25 202 657 833 640 814 679 835 658 196 638 812 837 204 637 810 205 660 839 636 808 713 206 662 841 806 6
63 209 210 804 665 212 211 802 666 213 214 800 707 215 798 901 216 709 203 903 796 217 220 898 794 905 222 223 224 896 792 908 225 226 8
94 227 790 911 228 229 788 913 893 231 218 786 914 891 234 784 235 916 889 236 782 237 887 920 781 238 239 922 886 779 221 242 923 884 2
43 778 244 246 882 925 776 247 248 254 880 774 927 232 878 772 258 929 259 876 770 930 260 261 874 768 932 262 873 264 766 934 266 871 2
33 764 198 936 268 761 460 869 272 938 271 219 867 457 273 940 267 275 866 453 276 277 941 278 279 864 284 449 285 942 287 862 288 436 2
89 944 290 860 291 293 425 946 294 858 253 274 337 423 948 856 208 339 297 108 409 950 854 338 250 952 191 281 850 680 296 954 184 848 6
73 169 846 676 735 990 153 845 736 988 341 85 151 843 992 986 682 148 714 721 152 842 983 105 711 141 724 188 981 342 95 715 201 979 139
 725 712 195 106 972 137 706 140 689 969 347 731 967 719 102 64 743 722 966 245 732 671 241 965 726 207 729 734 963 348 255 704 962 249
705 193 960 730 747 257 733 958 111 702 340 956 263 738 270 758 746 720 350 252 265 739 763 698 280 900 697 685 283 703 251 907 292 683
346 345 269 755 684 47 985 344 48 51 240 741 136 52 351 434 123 910 50 135 187 448 767 125 760 53 157 349 769 121 765 450 54 112 192 113
 771 451 155 717 97 452 56 775 58 16 777 59 780 115 49 60 994 783 61 63 62 690 787 66 789 65 186 692 495 785 68 795 70 55 797 174 497 67
 498 72 793 499 69 92 749 74 803 502 501 805 73 76 494 807 505 77 75 190 78 809 492 100 493 80 491 813 506 79 81 811 83 815 503 490 510
817 84 488 511 537 489 820 487 512 87 486 533 86 88 485 824 483 525 89 799 826 521 130 90 482 822 519 91 0 480 830 478 999 517 832 516 9
96 477 834 513 998 94 515 475 508 836 166 828 143 177 474 472 838 122 500 840 132 507 470 118 844 484 469 847 523 529 468 481 849 995 46
7 851 466 853 476 465 855 462 857 473 461 859 459 522 861 471 458 863 865 456 455 870 872 524 454 282 875 526 569 567 509 566 879 565 88
1 564 563 883 562 885 561 520 560 890 558 614 557 514 556 528 555 897 617 554 899 553 902 552 568 550 904 549 620 906 619 548 909 547 55
9 622 681 447 546 551 612 624 688 611 544 545 543 616 541 687 610 542 987 609 915 912 540 608 989 539 921 538 626 607 924 991 691 888 53
6 606 535 630 604 99 892 534 593 603 631 532 928 931 602 919 634 877 926 531 600 597 933 595 953 594 598 951 590 993 592 175 949 163 589
 587 947 945 586 584 179 103 943 957 583 581 939 937 12 582 935 723 710 578 716 579 740 627 686 575 577 744 757 632 674 576 759 675 670
677 678 574 648 650 667 823 672 659 256 621 343 669 230 978 625 718 976 651 727 628 852 970 107 138 728 762 791 773 496 518 974 801 504
570 479 618 613 71 463 955 571 918 464 527 82 868 286 997 917 573 572 635 189 530 abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$
```

ii) OpenMP Case2 1000 threads by using Compiler directive

Code:

```
#include <omp.h>
#include <stdio.h>

int main(int argc, char *argcv[]) {
	int pid = 0, numthreads = 1000;

	#pragma omp parallel num_threads(numthreads) private(pid)
	{
		printf("%d ", omp_get_thread_num());
		if (omp_get_thread_num() == pid) {
			printf("master");
		}
	}
	return 0;
}
```

Output:

```
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ gcc -fopenmp -o case2 case2.c
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ ./case2
0 1 4 2 3 6 5 9 10 12 7 15 14 13 16 17 20 11 8 18 22 21 24 19 25 26 27 28 23 30 32 29 33 31 35 34 36 38 39 37 42 43 40 41 44 45 47 46 50
 51 53 49 55 57 58 52 60 59 61 48 64 65 66 63 67 56 71 69 72 73 74 54 75 79 80 81 76 68 77 82 86 84 62 78 88 85 90 89 91 87 94 93 92 96
100 95 101 98 103 97 102 108 109 111 99 104 107 115 116 105 118 106 117 122 123 110 125 126 113 124 112 131 133 134 114 127 136 138 140
120 121 142 144 146 119 148 129 150 137 151 152 153 135 132 155 158 159 160 130 157 162 166 128 167 168 171 164 83 173 174 175 163 70 17
8 181 182 183 172 185 188 170 189 154 190 192 149 196 184 197 198 200 147 180 204 205 206 145 176 208 212 143 214 215 216 194 141 220 22
2 187 223 139 224 186 228 230 231 232 218 234 210 235 201 236 237 202 242 199 243 246 248 195 238 250 251 254 193 227 256 252 259 226 19
1 258 263 265 179 240 267 261 270 177 272 244 268 274 169 277 276 280 278 282 165 285 284 286 289 161 290 288 292 156 294 297 296 298 30
0 266 299 302 304 305 269 307 306 308 311 271 312 310 313 315 273 314 318 281 323 326 327 283 320 334 335 328 287 321 325 344 291 345 27
5 317 351 293 366 353 279 341 295 370 352 319 339 301 374 337 303 364 372 357 368 309 381 373 385 355 379 377 356 411 442 384 378 418 41
2 387 375 365 398 413 388 367 383 405 414 391 386 435 406 415 401 362 432 416 350 403 390 363 396 407 438 409 399 358 448 395 402 424 45
2 408 455 394 422 463 361 466 426 360 473 444 476 359 400 484 354 487 397 330 410 494 430 496 332 349 440 503 504 443 454 445 512 513 44
6 516 447 517 467 451 524 526 465 453 329 532 461 535 457 324 460 540 459 322 543 458 462 549 316 456 552 464 264 450 558 560 404 468 26
2 392 566 470 260 568 382 257 472 576 578 380 255 475 376 585 586 253 477 369 249 593 479 595 393 247 481 348 602 604 245 347 483 241 61
2 613 343 485 239 621 622 489 233 491 229 493 449 225 625 495 221 633 548 219 498 637 547 641 217 500 545 645 213 502 649 541 211 653 65
4 506 539 209 657 661 508 537 663 207 667 533 510 671 203 673 530 514 677 679 528 681 518 685 523 687 520 689 425 521 522 695 423 519 52
5 421 515 969 971 511 527 981 816 420 994 419 529 417 626 531 628 624 346 629 534 620 631 618 342 635 340 616 639 614 536 643 338 610 60
8 647 336 542 606 651 603 333 655 544 601 659 546 599 660 551 597 662 553 591 555 664 589 666 557 587 434 668 583 670 559 581 562 672 57
9 433 675 577 570 683 575 431 691 573 572 693 571 429 699 569 574 700 567 580 704 565 428 706 564 427 708 584 563 709 582 561 588 711 72
2 556 590 713 554 780 714 592 550 594 716 781 596 782 598 783 861 600 863 605 784 858 865 607 856 785 867 854 609 611 869 852 871 786 61
5 850 873 787 617 848 875 846 619 788 877 844 768 623 879 842 881 627 630 840 883 838 789 632 885 836 790 634 887 834 889 636 791 831 89
1 638 829 792 893 827 640 642 895 825 793 644 897 823 767 899 646 821 900 794 818 648 904 799 650 795 906 732 652 910 740 731 656 913 72
9 741 915 658 727 665 918 726 742 669 920 725 743 922 674 724 676 924 723 744 926 678 721 745 928 680 719 746 930 684 717 694 932 747 68
6 690 750 972 692 738 973 970 696 975 968 682 752 966 977 701 751 979 964 753 707 947 995 945 997 710 943 705 758 941 688 715 940 735 73
4 936 796 934 771 776 778 538 774 761 772 764 763 759 757 755 331 712 728 718 773 436 775 777 439 437 811 441 813 814 779 817 819 999 82
2 809 826 830 832 833 835 837 839 841 843 845 847 849 851 853 855 857 859 860 862 864 866 868 828 870 872 509 507 876 505 815 878 501 88
0 499 882 497 884 492 886 490 888 488 890 486 892 482 894 480 896 478 902 474 908 812 471 804 820 469 737 952 912 765 720 954 766 874 80
7 959 810 798 805 965 967 803 963 980 962 982 960 754 958 984 698 703 986 957 956 953 990 988 951 955 993 938 992 824 949 389 942 939 37
1 898 944 937 901 946 935 903 948 914 917 950 905 806 921 961 907 802 923 974 801 909 925 976 800 911 927 978 808 916 929 983 702 931 74
9 985 987 933 697 989 919 733 756 760 991 730 797 748 996 736 998 739 762 770 769 abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$
```

ii) OpenMP Case3 1000 threads by using Environment Variables

Code:

```
#include <omp.h>
#include <stdio.h>

int main(int argc, char *argcv[]) {
	int pid = 0;

	#pragma omp parallel private(pid)
	{
		printf("%d ", omp_get_thread_num());
		if (omp_get_thread_num() == pid) {
			printf("master");
		}
	}
	return 0;
}
```

Output:

```
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ export OMP_NUM_THREADS=1000
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ gcc -fopenmp -o case3 case3.c
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ ./case3
0 1 2 3 4 6 7 5 10 11 9 13 8 14 16 19 21 12 22 24 17 26 23 15 18 29 30 31 28 20 27 35 36 37 38 33 25 43 32 44 45 46 39 49 50 51 52 34 42
 57 41 58 59 60 40 66 67 68 69 56 64 71 72 73 74 55 63 80 54 81 82 83 62 53 61 89 90 91 92 48 47 98 99 100 101 84 103 104 105 106 94 79
111 112 113 93 114 78 120 88 77 122 123 124 87 76 129 131 133 86 132 75 137 85 70 140 141 142 145 65 148 149 150 152 153 121 154 155 119
 157 158 139 159 160 118 138 165 166 117 167 168 136 116 173 174 135 175 176 115 180 134 181 110 183 184 130 187 109 189 128 192 193 194
 108 197 127 107 200 201 202 126 205 102 209 208 125 210 97 213 217 96 218 219 95 221 222 223 224 229 170 171 231 232 233 169 236 172 23
9 240 164 241 177 243 247 163 178 250 249 251 162 179 254 161 257 258 182 259 262 156 267 266 268 144 270 146 275 274 147 276 143 278 28
3 151 284 285 286 289 288 290 291 235 234 293 294 230 295 296 279 228 301 277 303 304 305 227 273 308 311 226 312 313 272 315 225 318 27
1 320 321 322 220 269 325 216 328 329 265 330 333 215 264 336 337 338 214 347 253 350 349 348 212 355 263 358 363 211 342 366 367 207 37
2 373 374 206 375 380 381 204 383 384 387 203 391 392 256 394 199 399 400 346 401 198 345 196 408 344 411 195 413 416 418 417 191 422 42
4 190 426 427 431 188 432 433 435 186 439 440 441 185 444 448 340 449 450 453 456 457 455 459 460 463 352 467 468 470 351 476 478 477 25
2 341 486 485 339 487 248 495 335 246 496 497 334 245 503 505 506 332 244 510 331 513 242 514 516 327 238 521 524 523 326 237 530 532 32
4 261 533 539 540 323 343 542 255 319 548 549 552 260 317 556 558 316 562 564 567 298 572 571 299 575 297 300 579 580 292 302 584 587 28
7 588 306 592 593 282 590 307 281 599 309 595 280 310 597 606 314 601 607 609 605 611 613 614 603 404 620 621 622 627 628 629 377 631 63
5 376 636 371 633 600 378 641 370 637 643 379 369 648 649 639 651 382 368 650 645 656 365 385 658 652 647 662 364 386 654 665 653 388 66
0 670 361 657 673 389 360 669 678 659 362 390 359 671 661 393 688 357 675 395 663 356 695 677 396 667 354 679 397 703 704 668 353 681 39
8 672 711 687 674 655 676 720 689 760 680 754 691 682 733 755 693 684 715 697 756 716 686 462 699 757 751 461 690 701 758 692 458 683 77
1 759 694 454 685 761 773 696 452 713 762 403 698 451 763 775 776 700 447 764 779 446 783 774 736 781 445 769 794 735 443 785 702 442 73
4 787 803 438 806 732 790 809 437 731 792 436 814 815 795 730 434 821 797 728 822 430 799 727 829 429 831 801 729 428 837 805 840 425 72
6 843 807 423 724 848 851 421 811 721 420 856 813 858 722 419 817 723 866 415 819 725 871 414 874 823 719 412 878 881 828 768 410 830 88
6 887 409 766 832 407 753 892 894 834 406 752 899 836 405 739 902 746 838 906 777 748 841 912 913 778 744 845 920 780 747 847 770 782 92
5 772 784 767 891 786 928 884 889 893 788 936 882 895 402 789 922 880 943 897 791 879 948 950 901 877 903 875 905 872 862 953 954 907 87
0 873 860 959 909 868 960 854 876 865 911 963 852 883 966 863 915 850 971 888 859 917 846 976 890 978 857 919 844 896 855 985 984 842 92
1 898 853 839 989 923 991 900 849 835 927 904 997 464 833 908 465 961 827 910 957 914 824 472 968 466 916 955 473 965 975 918 474 977 95
1 924 967 471 475 979 949 926 973 479 981 947 930 480 983 945 932 481 987 941 934 482 993 939 938 483 995 937 940 484 935 488 942 933 50
8 944 931 509 946 507 929 511 504 952 502 956 501 958 561 500 962 563 560 499 964 559 565 498 557 970 566 494 555 972 568 493 554 974 56
9 492 553 980 570 491 551 982 573 490 550 986 574 547 489 988 576 546 545 577 578 544 820 581 543 818 582 541 816 583 538 812 585 537 81
0 586 536 808 589 535 804 591 534 594 802 531 596 529 800 598 528 798 602 527 796 604 526 793 608 525 992 610 522 994 612 520 615 996 61
6 519 825 617 518 826 618 517 619 998 515 623 512 624 625 718 750 714 737 712 717 738 710 742 709 749 708 741 707 745 706 743 705 740 66
6 765 664 861 864 634 638 867 632 640 869 630 642 885 644 626 969 646 999 990 469 abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$
```

2) OpenMP vector addition of 1 lack elements with serial and parallel with 2, 4, 8, 16, 32, 100, 1000 threads to select the fastest

Serial Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 100000

void assignRandomValues(int a[], int b[]) {
	for (int i = 0; i < N; i++) {
		a[i] = rand() % 1000;
		b[i] = rand() % 1000;
	}
}

int main(int argc, char *argcv[]) {
	int a[N], b[N], c[N];
	assignRandomValues(a, b);

	clock_t begin = clock();

	for (int i = 0; i < N; i++) {
		c[i] = a[i] + b[i];
	}

	clock_t end = clock();
	double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
	printf("Serial: %f miliseconds\n", time_spent);

	return 0;
}
```

Serial Output Code:

```
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ gcc -fopenmp -o vecADD vecADD.c
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ ./vecADD
Serial: 0.000718 miliseconds
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$
```

Parallel Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define N 100000

void assignRandomValues(int a[], int b[]) {
	for (int i = 0; i < N; i++) {
		a[i] = rand() % 1000;
		b[i] = rand() % 1000;
	}
}
int main(int argc, char *argcv[]) {
	int a[N], b[N], c[N];
	assignRandomValues(a, b);

	int p[] = {2, 4, 8, 16, 32, 100, 1000};

	for (int i = 0; i < 7; i++) {
		clock_t begin = clock();

		#pragma omp parallel private(i) shared(a, b, c)
		{
			#pragma omp for
			for (int j = 0; j < N; j++) {
				c[j] = a[j] + b[j];
			}
		}
		clock_t end = clock();
		double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
		printf("%d threads: %f miliseconds\n", p[i], time_spent);
	}

	return 0;
}
```

Parallel Output:

```
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ gcc -fopenmp -o vecADD vecADD.c
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp/Assignment$ ./vecADD
2 threads: 0.002884 miliseconds
4 threads: 0.000444 miliseconds
8 threads: 0.089227 miliseconds
16 threads: 0.020128 miliseconds
32 threads: 0.003935 miliseconds
100 threads: 0.008823 miliseconds
1000 threads: 0.078460 miliseconds
```

.’. 4 threads are best for my machine.

Extra: vector with 10 elements addtion to check OpenMP

Code:

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

#define n 10

int main() {
      double a[n], b[n], c[n];
      float startTime, endTime, execTime;
      int i;
      int omp_rank;

      startTime = omp_get_wtime();
      #pragma omp parallel private(i) shared(a, b, c)
      {
      #pragma omp for
      for (i = 0; i < n; i++) {
            omp_rank = omp_get_thread_num();
            a[i] = i * (rand() % 1000);
            b[i] = i * (rand() % 1000);
            c[i] = a[i] + b[i];
            printf("The value of a[%d] = %lf and b[%d] = %lf and result
            c[%d] = %lf done by worker Thread ID = %d\n", i, a[i], i,
            b[i], i, c[i], omp_rank);
      }
      }
      endTime = omp_get_wtime();

      execTime = endTime - startTime;
      printf("%f \n", execTime);
      return (0);
}
```

Output:

```
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp$ gcc -fopenmp -o sample sample.c
abhishek_n_n_20bce1025@ud:/mnt/D/ccpp$ ./sample
The value of a[5] = 3450.000000 and b[5] = 295.000000 and result c[5] = 3745.000000 done by worker Thread ID = 7
The value of a[9] = 5841.000000 and b[9] = 243.000000 and result c[9] = 6084.000000 done by worker Thread ID = 7
The value of a[6] = 2298.000000 and b[6] = 5316.000000 and result c[6] = 7614.000000 done by worker Thread ID = 1
The value of a[7] = 2702.000000 and b[7] = 2534.000000 and result c[7] = 5236.000000 done by worker Thread ID = 7
The value of a[4] = 3660.000000 and b[4] = 1968.000000 and result c[4] = 5628.000000 done by worker Thread ID = 7
The value of a[2] = 1586.000000 and b[2] = 842.000000 and result c[2] = 2428.000000 done by worker Thread ID = 7
The value of a[3] = 1620.000000 and b[3] = 1278.000000 and result c[3] = 2898.000000 done by worker Thread ID = 1
The value of a[0] = 0.000000 and b[0] = 0.000000 and result c[0] = 0.000000 done by worker Thread ID = 0
The value of a[1] = 172.000000 and b[1] = 736.000000 and result c[1] = 908.000000 done by worker Thread ID = 0
The value of a[8] = 6216.000000 and b[8] = 2680.000000 and result c[8] = 8896.000000 done by worker Thread ID = 7
time taken: 0.035156
```